UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

Degree in Computer Engineering

TRABAJO FIN DE GRADO

BACHELOR'S THESIS

CONTROL VISUAL DE ROBOTS QUE COMPITEN O COOPERAN

VISUAL CONTROL OF ROBOTS THAT COMPETE OR COOPERATE

Jaime Monedero March
Tutor: Luis Fernando Lago Fernández

Junio 2017
June 2017

VISUAL CONTROL OF ROBOTS THAT COMPETE OR COOPERATE

Author: Jaime Monedero March
Tutor: Luis Fernando Lago Fernández

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

June 2017

Acknowledgements

This project is the result of a busy year, all of the knowledge and know-how gained through my degree in *Ingeniería Informática* at the *Universidad Autónoma de Madrid*, an incredible amount of support, missed deadlines, met deadlines, a non-trivial amount of caffeine and more than one night in the lab watching robots bump into each other.

My mother's support is the one pillar that's helped me to get through that process, and her help through the hoops and the nights I've had to get through is the one thing that allowed me to overcome them. Thank you so much.

I would like to thank my teachers at the *Universidad Autónoma de Madrid*, and especially Luis Lago, my tutor in this project. His resources, guidance and direction were absolutely essential to the completion of this work.

The friends I've made on campus through these years have kept me active and grounded in the real world when I needed it, and that has made a huge difference in this part of my life. To classmates, a certain *WhatsApp* group and the *EPS* cafeteria alike, thanks.

And finally, to the rest of my friends, the ones that stayed up with me in completely different parts of the world: thanks for keeping me awake, working, not working, and sane.

I look forward to my next few years of education, and close this chapter with full knowledge that it was the correct step to prepare me for the rest of my career.

"Do what you will." — The Neverending Story, Michael Ende

Abstract

Abstract — This research project uses the software and hardware platform developed in a previous work as a starting point, improving on it in order to later utilize it to research a related problem. The goals of this work are threefold: improve the control algorithm of the associated robots in order to enhance trajectory accuracy, expand the platform to enable it to control more than one robot, and use this expanded platform with two robots to test simple movement strategies for a specific case within the pursuit-evasion problem.

Firstly, we examine the algorithms that determine the robots' movement dependent on its position and orientation relative to the desired position. With these as a starting point, we create a more complex algorithm that improves the accuracy of the robot's trajectory towards its destination by reducing counter-productive movement. We also test several modifications to the system to attempt to improve the tracking of the robot and in doing so enhance the trajectory accuracy as well, and conclude that the drawbacks they introduce make them unsuitable for implementation into the system. Secondly, we modify the software platform in order to add the capability to control more than one robot at once.

Finally, we use the platform in conjunction with two robots to test the effectiveness of several simple strategies in a single-pursuer, single-evader, continuous search space variation of the pursuit-evasion game. An extensive battery testing is performed, comprising a total of 4030 individual tests. The optimal parameters for each of the strategies are identified, and we perform a short qualitative analysis of the effectiveness of each pursuit and each evasion strategy in relation to the others.

Key words— Computer vision, visual servoing, OpenCV, Python, Arduino, Bluetooth, search, pursuit-evasion, lion and man, homicidal chauffeur

Resumen

Resumen — Este proyecto de investigación utiliza la plataforma hardware y software desarrollada en un projecto anterior como punto de partida, mejorándola para más tarde utilizarla en el estudio de un problema relacionado. Los objetivos de este trabajo son tres: mejorar el algoritmo de control de los robots asociados para aumentar la precisión de sus trayectorias, expandir la plataforma para permitir que controle más de un robot, y utilizar esta plataforma expandida con dos robots para probar estrategias de movimiento simples en el contexto de un caso específico del problema de persecución-evasión.

En primer lugar, examinamos los algoritmos que determinan los movimientos de los robots dependiendo de su posición y orientación relativas a la posición deseada. Con éstos como punto de partida, creamos un algoritmo más complejo que mejora la precisión de la trayectoria del robot a su destino reduciendo movimiento contraproducente. También probamos varias modificaciones al sistema para intentar mejorar la localización y seguimiento del robot y de esta manera mejorar también la precisión de su trayectoria, concluyendo que las desventajas que introducen los determina insatisfactorios para su inclusión en el sistema. En segundo lugar, modificamos la plataforma de software para añadir la capacidad de controlar más de un robot al mismo tiempo.

Finalmente, usamos la plataforma en conjunción con dos robots para probar la efectividad de varias estrategias simples en una variación del juego de persecución-evasión con un solo perseguidor, un solo evasor y espacio de búsqueda continuo. Una extensiva batería de tests es efectuada, comprendiendo un total de 4030 tests individuales. Los parámetros óptimos para cada una de las estrategias son identificados, y efectuamos un corto análisis cualitativo de la efectividad de cada estrategia de persecución y evasión en relación con las otras.

Palabras clave — Visión por ordenador, control visual de servos, OpenCV, Python, Arduino, Bluetooth, búsqueda, persecución-evasión, león y hombre, conductor homicida

Glossary

Arduino A company that designs and produces single-board microcontrollers to allow computers to interact with the physical world. By extension, the name refers also to the family of boards it manufactures. Depending on the board model, they are equipped with interfaces such as USB, Bluetooth and GPIO pins, in order to communicate with computers and with low-level electronic components. (http://www.arduino.cc/). 33

Bluetooth A technology and communications protocol used to create wireless Personal Area Networks (PANs) to allow devices within close proximity to communicate data. It is widely considered to be the standard for short-distance wireless communication, and is most commonly used on cell phones. (https://www.bluetooth.com/). 33

OpenCV An open-source library designed to provide tools to tackle the problem of real-time computer vision. The library is free for use across platforms, and provides interfaces for multiple programming languages. (http://opencv.org/). 19

Python A general-purpose high-level interpreted programming language, with a design philosophy which prioritizes code readability and a compact syntax. (https://www.python.org/).

Acronyms

CAMShift Continuously Adaptive Mean Shift 26, 34

FPS Frames Per Second 30, 32

IBVS Image Based Visual Servoing 24

PBVS Position Based Visual Servoing 24, 26

Table of contents

List of figures

List of tables

1 Introduction

The goal of this project project is the enhancement of a given software platform for the control of simple robots by means of an overhead camera, and the use of said platform in the particular application of a specific case in the field of search problems.

1.1 Motivation

The use of physical agents in pursuit-evasion problems in conjunction with visual servoing offers the possibility to examine the viability of said combination of technologies in real-world applications.

In [1], Iván Márquez Pardo has created a software platform that allows the control of simple robots through the use of visual feedback provided by a stationary camera. It is based on a very simple and affordable robot and camera setup, allowing the robot to be controlled in a flat environment without obstacles, and within a limited workspace. Through the use of OpenCV, it performs lens distortion correction and perspective correction on the captured images, making it possible for the setup to behave as if it was observed directly from above by an overhead camera, even if the camera is not directly above the workspace.

The flexibility of this platform makes it a convenient system to perform testing within an academic environment. However, the control of a single robot having been proven viable in the previous work, it was decided to expand the platform to allow the use of more than one robot, in order to be able to use it for problems that would examine the interaction between two or more robots, in the framework of either a competition or cooperation setting.

The choice of the pursuit-evasion problem was made taking into account its relative simplicity in relation to other two-player problems, requiring only a minimal amount of shared information between the two players. This allows the two sides (and thus, the controllers of each robot) to adapt closely to the design of the platform.

1.2 Scope

This work studies the possible improvements to an already existing software and hardware platform for visual servoing, as well as the testing of possible solutions for a specific problem in the field of search theory.

Our approach to the improvement of the software seeks to expand on it rather than create a different version of it, the viability of the software platform having already been validated in [1]. The tracking algorithms are considered satisfactory, but the robot control algorithms are examined in order to enable the robot to take more direct trajectories to its destination points. In order to improve the performance of the system, the possibility of the use of a different camera setup is raised, and some qualitative testing is undertaken with a phone camera, before concluding that the performance of said phone camera solution is not comparable to that of the initial system.

The discussion of the suitability of this platform for the study of the proposed search problem falls outside the scope of this work. Furthermore, the use of this platform to study other problems involving cooperating or competing entities is suggested in section 7.1 (future work), but not examined in detail in this paper.

1.3 Project approach and methodology

In order to effectively study the proposed pursuit-evasion problem, it is first deemed necessary to adjust the robot control algorithms and overall performance of the system, allowing for a more effective testing scenario. The modifications introduced into the system to improve its behavior are examined through qualitative testing, and considered for implementation or abandon based on its increase or decrease of the system's performance.

In the pursuit of the objective of expansion of the system to allow for control of more than one robot, a simple, modularized code redesign is implemented. This new design allows for each robot to move following the trajectories determined by its own mode of operation. In these trajectories, the modified variable is the aim point towards which the robot will attempt to move based on the improved control algorithms.

Finally, several simple movement strategies are designed, informed by the research done on the continuous space pursuit-evasion problem. These strategies are tested against one another in an exhaustive battery testing, and the results are analyzed both qualitatively and quantitatively to determine the most effective pursuit and evasion strategies, as well as the optimal parameter combinations for each of them.

1.4 Structure of this document

This document is organized as follows. In section 2, we review relevant work in the pursuit-evasion problem, particularly in the area of low-maneuverability, continuous-field pursuit-evasion; as well as in the visual servoing field, particularly in the areas relative to the implementation of the existing system. Section 3 details the methods explored to improve the precision of the trajectories performed by the robots by means of modifying the control algorithm and the physical components of the system. In section 4, we explain the most prominent challenges to consider when modifying the design of the system to accommodate more than one robot. Section 5 is dedicated to defining the scope of the area of the pursuit-evasion problem that is tested in this project. Section 6 presents the results collected through experimental testing and a data-based comparison of the pursuit-evasion strategies tested. Lastly, section 7 presents the conclusions of the research and proposes directions which future work on this subject could progress towards.

2 State of the art

In the last decade, the use of robotics for personal, work and research use has become increasingly accessible and inexpensive, enabling its use in research to be much more widespread, and advancing related fields at a faster pace. As more advancements are made in robotics, they allow for other fields to use robots as tools to explore new possibilities.

One such advancement in the field of robotics is that of visual servoing: the use of cameras and visual feedback to control robots in relation to their environment. This allows the system to gain information not only from its internal programming and received data but also from the external environment in which the robot operates, in real time [14]. The applications for a system with these characteristics encompass examples like the use of robots to perform surgery [22], to improve precision and allow adaptability of robot systems in factories [23], or to aid in navigation of urban environments to complement GPS tracking [24].

2.1 Visual servoing

Visual servoing refers to the use of a camera to provide visual feedback in order to inform the behavior of a robotic system. The captured images are analyzed in real time, providing information about the position and orientation (also called pose) of components of the robotic system, as well as information about the state, topology and features of the environment the system operates in. [14]

A superficial taxonomy of the visual servoing problem follows; for a more in-depth summary, the reader is invited to read [1], in which visual servoing and the different solutions implemented in that field are explored and explained in more detail.

Depending on the camera position, the visual servoing field can be broken up into three distinct approaches:

- Eye-to-hand [16]: In this approach, the camera is mounted on a fixed point, with a complete or partial view of the workspace the robot operates in. This approach provides information about the environment at large and the positions and features of both the robotic components of the system and those of the environment, while not being as precise as other approaches due to the distance from the system necessary to encompass the workspace. This solution is most useful in problems that do not necessarily require extremely precise information but do require an ample breadth of it.
- Eye-in-hand [15]: In this approach, the camera is mounted to a moving element of the robot itself. While the movement can entail not being able to cover the entire workspace, the ability to move the camera closer to points of interest increases the accuracy and precision of the information gained from the visual feedback.

- Mixed [17,18]: By combining the two previous approaches, we can use a two-or-more-camera setup to enable each camera to minimize the shortfalls of the other.

Based on how the captured image is processed and interpreted, the control algorithm can fall under three categories:

- IBVS (Image-Based Visual Servoing) considers the images as a whole, without separating it into regions or isolating special features. The procedure for this control scheme is to send movement instructions in order to reduce the error between the desired image parameters and the current captured image [25].
- PBVS (Position-Based Visual Servoing) identifies features in the image and correlates it to a known structure, calculating through it the pose (position and orientation) of the robot, and sending movement instructions in order to reduce the error between the desired pose and the current pose [25].
- Mixed: An approach that uses both of these to complement each other, or that switches from one to the other based on appropriate criteria [26].

If the control algorithm is PBVS or hybrid, the program then uses tracking algorithms to track the position of these features across frames of video, updating its information and recalculating the error over time [27].

2.2 Pursuit-evasion problem

Pursuit-evasion describes a type of search problem in which a searcher or searchers attempt to find a target or targets in a search space. Several characteristics distinguish between search problems in general, of which pursuit-evasion problems are a subset [2]:

- Adversarial targets / Non-reactive targets: This distinction capitalizes on the objectives of the targets: non-reactive targets are the defining characteristic of what is called “one-sided search”, so called because the searchers are the only ones with an objective (reducing the time to capture), in which the targets are randomly placed, move randomly, or move deterministically without reacting to the searcher’s movement. In contrast, adversarial targets are the defining characteristic of the pursuit-evasion problem: targets will attempt to maximize the time to capture as the searchers will attempt to minimize it. In the latter case, this field overlaps with the field of game theory, given that each of the parties (searchers and targets) pursue opposing objectives and must execute strategies to overcome the other side.
- Mobile targets / Stationary targets: While the latter category is most often found in one-sided search, adversarial versions of the problem have also been studied, in which the adversary chooses the target(s) initial positions in such a way as to maximize the search time.
- Number of searchers / number of targets: While some problems are defined by the existence of only one searcher or one target, other problems are geared towards finding

the minimum number of searchers or targets necessary to guarantee capture of all targets or to guarantee indefinite evasion of at least one target [6,7].

- Definition of "win": Some examples of capture conditions include location of the target by line of sight [2], movement of the target to a determined area [5], occupation of the same node on a graph (if the space is discrete) [20] or clearance of a proximity threshold.
- Continuous space / Discrete space: Discrete space is the defining characteristic of "graph search", in which the search space can be represented as a graph and all movement is constrained to movement between the nodes of that graph [8]. On the other side, continuous space (also called geometric space) considers the entirety of a defined area (bounded or unbounded) to be valid positions for both parties [4].
- Continuous time / Discrete time: In discrete time, the movement is performed in "rounds", with the targets and the searchers moving one after the other. In continuous time, however, all parties move simultaneously. In some problems, (e.g. Lion and man in a finite arena), this difference can determine whether a strategy that guarantees capture or evasion exists [2].
- Finite search space / infinite search space: The search space can be defined as having boundaries or not; and further than that by the shape of those boundaries.
- Perfect knowledge / Imperfect knowledge: A game with perfect knowledge implies that both the searchers and the targets always know the position of all elements in the search space, as well as its boundaries and in some cases the direction and velocity of every party. Imperfect knowledge games may impart constraints such as line-of-sight knowledge (either of the parties can only know the other's location if they are within a defined distance and/or if there are no obstacles or boundary lines between them) or unknown search space (which must then be mapped out as it is explored). Most search problems consider the opponent's strategy to not be known to either of the parties, but some have studied behavior to counter known strategies [3].
- Movement constraints: Constraints on movement other than the bounded or unbounded nature of the search space can include constrained maximum speeds or limited maneuverability (when changing direction or velocity) [9]. Any of these can be different between the two sides, as shown in problems such as the homicidal chauffeur [19].

2.2.1 Well-known examples in search

The following are several well-known and researched problems in the field of search, each considering a different combination of the characteristics described in the previous section [2]:

- Data search algorithms [21]: Within this problem we find information-searching algorithms like binary search, A* or brute force search; most commonly used to find data within a discrete, finite search space. The target is single and stationary, and is placed at random or following a non-reactive pattern (e.g. for a binary search, data must be ordered, and it will be ordered in the same way regardless of the searcher's behavior).

- Cops and robbers [20]: A discrete-space problem in which a robber (target) attempts to avoid capture by one or more cops (searchers), commonly in a discrete search space. The objective of these types of problems is usually to find the minimum amount of cops necessary to guarantee capture of the robber regardless of initial placement for a given graph or class of graphs [citation needed]; or to determine the classes of graphs in the first place.
- Lion and man [13]: Contrary to the cops and robbers problem, this problem considers an adversarial search in a continuous space: a lion (searcher) must reach a person (target) in a finite or infinite "arena". Most researched variants of this problem tend to distinguish themselves by the search space, such as a finite circular arena, an infinite arena with two boundary walls (first quadrant) or a polygonal arena.
- Homicidal chauffeur [19]: This problem can be considered a variant of the Lion and man problem: a driver in a car (searcher) tries to run over a pedestrian (target) in a continuous search space. In this variant, the car has a higher speed than the pedestrian but also lower maneuverability: the pedestrian is slower but can make sharp turns and vary their speed in a shorter amount of time than the car is able to.

2.3 About this project

This project touches upon both the visual servoing field and the pursuit-evasion field. It builds upon the hardware and software resulting from the project Visual Control of a Mobile Robot By Means of an Overhead View by Iván Márquez Pardo [1], improving the control of the robot described in the project and adding capabilities for more than one such robots to be controlled by the software.

The setup created in that project was based on a single eye-to-hand camera, with PBVS analysis based on the CAMShift tracking algorithm [28]. These visual servoing choices were not modified, this study simply building upon the existing system.

The application we have chosen to focus on is the testing of simple strategies in the area of the continuous-field pursuit-evasion problem. The setup involving two robots allowed us to test different movement strategies in a single-pursuer single-evader scenario, with maneuverability constraints in place and similar speed constraints on both sides. Due to the nature of the physical workspace, it considers a bounded (rectangular), continuous search space in continuous time. Both robots are given perfect knowledge of the other's pose (location and orientation), but not of each other's pursuit or evasion strategy.

3 Control mechanism improvement

The previous work in [1] was based on the use of a control scheme that was as simple as possible: the objective of that work was simply to validate the viability of the system and show that it worked along with providing a sample implementation. In this work, we decided to refine the control scheme in order to make the movement more precise and efficient.

3.1 Movement strategies

Part of the work done by Iván Márquez Pardo was to create movement strategies to allow the robot to move as efficiently as possible. The result of his work was two separate weighted strategies with highly optimized weights calculated through extensive testing. In this work, we developed three more strategies to allow the robot to move in a more optimized way. Strategies 1 and 2 described here were developed in [1], and details of calculations associated with them can be found in that work. Strategies 3, 4 and 5 were developed over the course of this project.

3.1.1 Strategy 1

The robot calculates its angular error respective to the objective position. If the error is below a threshold (20º), the bot moves directly forward, at a speed determined by its distance from the objective position (its positional error). Otherwise, the bot stops completely and corrects its angle so that it faces towards the objective position.

This strategy results in a trajectory that diminishes the chance of moving away from the objective at any point. However, it results in a high frequency of stops for the robot to correct its angle, especially if the two motors aren't correctly calibrated: in this case the robot will tend to drift to the side when both motors are set to move at the same speed. When the robot is close to the objective point, the number of required stops grows at a high rate, since smaller forward movements are sufficient to make the angular error increase.

3.1.2 Strategy 2

The robot corrects both its angular error and its positional error simultaneously, moving forward at a speed determined by its positional error while turning to face the objective.

This strategy does not take into account the angular error before setting the robot's forward speed. Thus, if the robot starts its movement facing away from the objective point, it will move away from it. This results in inefficient trajectories which, while taking less time than the ones generated by strategy 1, describe wide curves and cover a large amount of unnecessary distance.

3.1.3 Strategy 3

Strategy 3 is a combination of strategies 1 and 2, as shown on figure 3.1. If the angular error is above 30º, the positional and angular speeds are determined by strategy 1, turning to face the objective before making any forward movement in order to avoid the runaway effect caused by strategy 2. Otherwise, if the robot is facing towards the objective (less that 30º of angular error), strategy 2 is used to let it move towards it while continuously correcting its angle without stopping, avoiding the frequent stops caused by strategy 1. Once the robot's positional error drops below 75 pixels (5 times the positional tolerance, 15 pixels), the robot uses strategy 2 in order to ensure its steady movement towards the aim point, thus preventing the choppy movement close to the objective that caused large delays when using strategy 1.

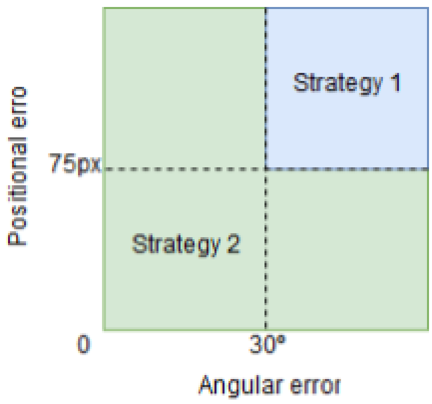

Figure 3.1: Visualization of strategy choice

3.1.4 Strategy 4

This strategy functions in the same way as strategy 3, but keeps the angular and positional speeds at their minimum. The servos only turn when their speed is set to a value of ±75 around their resting point, so the strategy chooses angular and positional speeds to ensure the servos will turn as slow as possible. The angular error thresholds in this strategy are set: when the angular error is lower than 10º, the robot moves in a straight line; when the angular error is higher than 40º, the robot stops its forward motion and turns; otherwise, it keeps its forward motion while turning at the same time.

3.1.5 Strategy 5

This strategy functions in the same way as strategy 4, but allows for the threshold values for the angular error to be passed in as parameters. In this way, we can use the test routines programmed in [1] to test different threshold values for the angular error. The cursory testing conducted for this purpose is explained in section 3.2.1.

3.2 Other methods

During our research, no changes were made to the manner the robot is tracked or the manner its pose is calculated. We refer the reader to the work done by Iván Márquez Pardo in [1] for a detailed explanation of the tracking process.

The image from the IP camera is transmitted in JPG format, at a relatively low resolution of 640x480. At this resolution, and taking into account JPG compression, the image is not sufficiently crisp to calculate the robot's orientation respective to the workspace to a high degree of accuracy. Two methods were explored to improve the image quality and with it the tracking accuracy: increasing the resolution of the IP camera and using the camera of a smartphone.

3.2.1 Increasing resolution

The D-Link DCS-3110 camera, used in the setup inherited from the previous work, allows for the resolution to be doubled from 640x480 to 1280x1024. In the previous work, the decision to set the resolution to the former value was motivated by two reasons:

- In this camera model, the framerate is limited to 8 FPS if the 1280x1024 resolution setting is chosen, as opposed to the 640x480 resolution setting, in which a framerate of 30 FPS is available.
- The processing cost for this higher-resolution image is heightened due to the larger size of the capture. On the setup used in the laboratory, this higher cost caused the computer to introduce processing lag, effectively reducing the framerate to a value lower than the already-reduced 8 FPS.

Drawbacks

At the higher resolution, the IP camera introduces a large amount of delay, upwards of half a second, which constitutes a large difference compared to the negligible delay introduced in the lower resolution. This delay causes the movement instructions to be sent in response to a processed capture that is no longer representative of the real world. Due to this delay, the robots consistently overshoot their target positions, as well as their target angles. This causes the trajectory to become wildly inaccurate. In many cases, the robots don't reach their destination, instead being caught in a loop, attempting to turn towards their aim point, overturning, and repeating the process in the opposite direction endlessly without making any forward progress.

In addition to the inaccurate trajectories, a large amount of lag was introduced, further exacerbating the delay, due to the computer having to process a much larger image. This is explained further in point 3.3.

Drawback mitigation strategies

In order to mitigate the effects of the lag, the speed of the robots was lowered to its minimum by means of strategy 4. Slowing down the movement of the robots led to the movement made in the time between reaching the objective values and receiving the feedback that confirms it being lowered to a smaller amount, thus overshooting the targets by a smaller distance. This allowed the trajectories to reach the objective points in a more consistent manner.

Strategy 5 was written in order to conduct cursory testing, in hopes of finding an optimal combination of threshold angles. The following values were tested:

- Lower threshold: [10, 20, 30, 40]
- Upper threshold: [40, 55, 70, 85, 100, 115, 130]

All values for the lower threshold were tested with all values for the upper threshold. The most efficient combination (lowest mean time and standard deviation of time) was identified as (10, 55). However, more extensive testing was not conducted due to this approach being discarded. While the trajectories were more precise than the ones resulting from using strategy 3 at the higher resolution, they were not more precise than the ones resulting from using strategy 3 at the lower resolution. In addition to that, the time to reach any of the aim points was drastically increased in comparison to the times achieved before the resolution change.

3.2.2 Phone camera

Due to the loss of precision introduced by the delay of the IP camera not being mitigated to a satisfactory degree, tests were undertaken to establish the benefits of using a phone camera instead.

The phone used was a Huawei P8 Lite, running Android 6.0, connected to the computer via USB. The interface used to access the phone camera was set up using the Droidcam project [10], consisting of:

1. A service running on the Ubuntu operating system
2. A connection to the phone based on ADB (Android Debug Bridge, part of the Android Developer Tools) [11]
3. A companion app running on the smartphone

This setup still introduced an unsatisfactory amount of delay. Although the delay was consistent across different resolution settings, and was determined to be lesser than the IP camera's high resolution delay, it still remained higher than the IP camera's low resolution delay, resulting in the problems described in the previous section.

3.3 Processing lag

In addition to the lag introduced between the camera capture and the computer, lag can also be introduced during the processing of the image done by the tracker module. This lag causes the same drawbacks as the capture lag, leading the trajectories to lose a high amount of accuracy and jeopardizing the ability to reach the objective points.

During the course of this research, two main sources of processing lag were detected:

1. The higher size of the image associated to a higher resolution capture entailed a longer processing of the image when applying the transformations to correct the lens distortion

and perspective tilt. Due to the longer processing time, the number of processed frames per second dropped drastically, from its maximum of 8 fps to 3 fps. This was mitigated in part by skipping some of the steps in this image correction process: skipping the perspective correction affine transformation diminished the processing burden and raised the frame rate back to 7.5 fps.

2. The program keeps track of the trajectory the robot has followed and draws it on the frame in the form of a trail of points. Regardless of image capture resolution, if the program is left running for enough time, the amount of points that must be drawn will become high enough for the trajectory drawing step to introduce significant lag in the frame processing. This problem was solved by limiting the trajectory points to be drawn to only the last 200 points, causing the trajectory shown on the feedback feed to only encompass the most recent history, and letting the older trajectory points disappear from the image.

3.4 Conclusions

After comparing the performance of the system with these modifications, it was determined that only the strategy improvement modification yielded a significant improvement to the precision of the robots' trajectories, with little to no drawbacks. In consequence, the only modification maintained through the rest of the research is the use of strategy 3 over the previous strategies.

4 System expansion for multiple robot control

4.1 Code design

The high-level design of the robot tracker and controller programs was maintained through the code rewriting in this work.

The tracker and the controller modules are executed in separate processes, sharing a pipe to send robot pose information and basic control commands. Following this decoupling, the robots were modeled in two different ways, each related to the way either module interacts with the robots:

- Tracker_Robot holds information about the position and orientation of the robot at any given time, the color of its markers, and the mode that each particular robot is running, along with organizing some robot-specific methods in order to make the class more opaque.
- RC_Robot also holds information about the robot's pose, as well as the pipe that connects the robot to its respective Bluetooth port. Most of the methods and strategies related to the robot movement are moved into this class, as well as some of the logging tasks.

Thanks to the changes introduced, it was possible to merge all of the three applications from the previous work into a single generalized method in both the Tracker class and the Robot_Controller class, while allowing the Tracker_Robot and RC_Robot classes to determine the trajectories depending on the mode that was activated. The testing routine was completely revamped, modified to fit the tests that would be done, explained in section 5.

4.2 Hardware considerations

The great majority of the hardware used in this work is identical to the setup used in Iván Márquez Pardo's work. We refer the reader to [1] for specific details on the robot's hardware and design.

The only diferring element in the hardware setup from the previous work to this iteration is the addition of a second robot. The second robot was printed following the same specifications and design as the one used in [1], and was outfitted with the same BQ ZUM BT-328 Arduino-compatible board, and the same SpringRC SM-S4303R servos.

4.2.1 Tracking considerations

An immediate obstacle to overcome is the fact that for the tracking to be done with a minimum measure of accuracy, the colors for the color markers need to be different and distinguishable from one another, both by the CAMShift algorithm used to find the position color tags and by the mask-contour-based algorithm used to find the orientation color tags. It is necessary to select four color markers that are different and saturated enough to not be mis-identified, and not dark enough or light enough that they will be washed out by the camera image.

Yellow has been observed to be prone to washing out and becoming too desaturated, and any similarly light colors would not be recognizable because of the low saturation. Most shades of purple are either too dark or too light, the desaturation making them wash out easily as well. An added challenge is presented the fact that the floor of the workspace is mostly blue, thus making most shades of blue liable to merging with the background.

In the end, the colors chosen were green and orange for robot 1, and hot pink and cyan for robot 2. The cyan was light and saturated enough to not blend in with the background of the workspace, and though red has been found to work well with the camera, the pink and orange were chosen to be as far apart from each other as possible.

4.2.2 Robot control considerations

In the previous work, the existence of obstacles was not considered in the design of movement strategies. However, the introduction of a second robot creates a moving obstacle in the workspace for both of the robots.

Though pre-existing applications (fixed point trajectory, mobile trajectory, ARDUAMBOT platform integration [citation needed]) were not modified to take obstacles into account, the testing procedure created for the pursuit-evasion strategy testing (described in section 6) was designed with this feature in mind. During the testing sequence, after each test, the robots return to their starting positions to begin the next test. A simple algorithm was implemented to prevent them from crashing into each other:

1. If the bots are close enough to bump into one another (75 pixels $\approx$ 12.8 cm), they set their aim points directly away using the blind evasion strategy, described in section 5.2.2.
2. If the bots are more than 195 pixels apart ($\approx$ 33 cm), they move directly towards their destination point.
3. If the bots are between 75 and 195 pixels apart, each bot calculates whether it needs to go around the other robot to reach its destination point. If the other robot does not interfere with the current robot's trajectory, the bot will move directly towards its destination point; otherwise, it will use the perpendicular-to-pursuer evasion strategy described in 5.2.2 to move clockwise around the obstacle. The other robot is considered an obstacle if:
    a. The current robot is closer to the other robot than to its destination point
    b. The angle between the destination point, the current robot and the other robot is less than 75º

Despite this algorithm, after certain tests, the robots would be too close before starting to pull apart, and would interlock with each other, both trying to move away from each other but doing so by attempting to turn while going forwards, and preventing each other from moving. In order to prevent this, a simple piece of card stock was added, covering the space between the wheels and the body, as shown on figure 4.1. After the addition of this bumper, the problem did not present itself again.

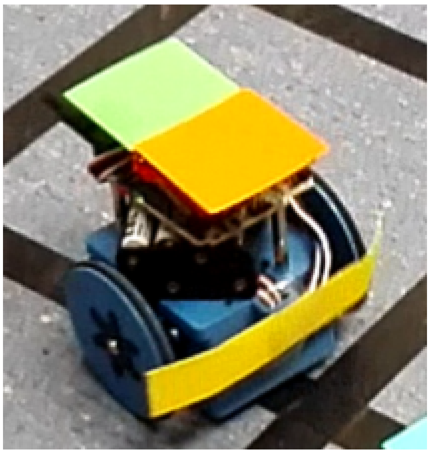

Figure 4.1: Robot with "bumper".

5 Application in the pursuit-evasion problem: strategies for pursuit-evasion

5.1 The pursuit-evasion problem as explored with visual servoing

Within the taxonomy of search problems described in section 2, as informed by the study in [2], we find a certain subset of the adversarial search problem to be of particular interest when applied to the visual servoing problem: a variant on the homicidal chauffeur problem [19].

Our visual servoing setup (with two robots) lends itself well to it: due to the constraints of the physical world, the problem is tested in continuous space and continuous time, in a finite search space equivalent to our work space. A one-pursuer, one-evader setup with a mobile target is the most interesting way of using our tools to investigate this problem.

A variant on the homicidal chauffeur is proposed: instead of one pursuer with high speed and low maneuverability and one evader with high maneuverability and low speed, we procured two “cars”, both with the same kind of maneuverability and speed constraints, given by their movement strategies as described in section 3.

We chose to use an empty workspace without obstacles, to keep the problem as simple as possible. Additionally, every robot has perfect knowledge of its own pose and of its opponent’s pose, unrestricted by line of sight or distance. However, it does not know what its opponent’s strategy is.

A capture is recorded when the distance between the two robots is reduced below a threshold, representative of the pursuing robot reaching the evading robot. This threshold was set to as close of a distance as possible without the robots striking each other: 75 pixels ($\approx$ 12.8 cm)

5.2 Description of the strategies

All the pursuit and evasion strategies were developed over the couse of this project. Due to the fact that our intention was simply to demonstrate the platform’s capabilities, the strategies were intentionally designed to be simple, and were not based on any other works. When discussing the following strategies, P refers to the pursuing robot and E refers to the evading robot. All of these strategies operate by setting the aim point towards which the robot should move, determining its trajectory and dynamically calculating its target position without modifying its movement strategy (i.e. the way it moves to reach the target point, described in section 3.1).

5.2.1 Pursuit strategies

Below are described the different strategies devised for the pursuing robot to attempt to reach the evading robot, along with the parameters that modify the performance of each strategy.

Direct chase

P sets its aim point to the current position of E. It can be simplified as the interception strategy described in the next section, with a *dist* parameter of 0.

Interception

Using the position and the orientation of E, P sets its aim point in front of E, in hopes of intercepting its trajectory. The distance ahead of E to which the aim point should be projected is given by the *dist* variable. The aim point is calculated in the following manner:

$$x_a = x_e + cos(\widehat{e}) * d \qquad y_a = y_e + sin(\widehat{e}) * d$$

where:

- x_a and y_a are the coordinates of the point the robot will aim towards.
- x_e and y_e are the coordinates of the evading robot's location.
- $\widehat{e}$ is the angle towards which the evading robot is facing, giving the direction that will be used to project the aim point.
- d is the projection distance, given by the dist parameter.

Parameters

- *dist*: Distance (in pixels) to project the point ahead of E.

Dynamic interception

As in the interception strategy, P sets its aim point a certain distance ahead of E, taking into account E's position and orientation. However, as P's position approaches E, the distance to which the point is projected diminishes, causing the aim point to get closer to E the closer the two robots are. This reduction is calculated in the following manner:

$$r_b = \frac{d_{PE}}{\sqrt{480^2+640^2}} \qquad d_{f1} = r_b \cdot d_b \qquad d_{f2} = \sqrt{r_b} \cdot d_b \qquad d_{f3} = {r_b}^2 \cdot d_b$$

where:

- d_{PE} is the Euclidean distance between P's position and E's position, represented as 2D coordinates on the transformed image.
- $\sqrt{480^2 + 640^2}$ is the maximum possible distance between two points (Euclidean distance between the opposite corners of a rectangular workspace of size 640x480).

- r_b is the base ratio, calculated by dividing the distance between the robots by the maximum possible distance. Depending on the function chosen, a different modifier is applied to it (linear, square root or quadratic; the specific process is described below) before it is used as a coefficient to multiply the base projection distance.
- d_b is the base projection distance, given by the *dist* parameter.

The base ratio is modified in one of three ways, chosen by the *coef_func* parameter, before using it to multiply the base distance:

1. Linear: d_{f1} is the final distance that the point will be projected ahead of E, said distance changing linearly.
2. Square root: d_{f2} is the final distance that the point will be projected, said distance changing faster the closer the robots are to each other.
3. Quadratic: d_{f3} is the final distance that the point will be projected, said distance changing faster the further the robots are from each other.

The aim point is calculated in the same way as in the previous strategy:

$$x_a = x_e + cos(\widehat{e}) * d_{fi} \qquad y_a = y_e + sin(\widehat{e}) * d_{fi}$$

Parameters

- *dist*: Base distance (in pixels) to project the point ahead of E.
- *coef_func*: The values of this parameter each designate one of the three reduction functions:
    1. Linear
    2. Square root
    3. Quadratic

5.2.2 Evasion strategies

Below are described the different strategies devised for the evading robot to attempt to avoid the pursuing robot, along with the parameters that modify the performance of each strategy.

Blind

E calculates the PE vector and sets its aim point to be at the destination point of the PE vector when its origin is set at E. This results in an aim point that is diametrically opposed to P relative to E:

$$x_a = x_e - (x_p - x_e) \qquad y_a = y_e - (y_p - y_e)$$

where:

- x_a and y_a are the coordinates of the point the robot will aim towards.

- x_e and y_e are the coordinates of the evading robot.
- x_p and y_p are the coordinates of the pursuing robot.

Naïve

E sets its aim point in the direction opposite to P (relative to E), setting it further away from E the closer the two robots are. Because the movement strategies increase the robots' speed when their aim point is further apart, E increases its speed when the other robot is closer; as opposed to the previous strategy, within which E would decrease its speed the closer the pursuing robot was. The aim point is calculated in the following manner:

$$x_a = x_e + (\frac{|x_e - x_p|}{d_r}) * d_p \qquad y_a = y_e + (\frac{|y_e - y_p|}{d_r}) * d_p \qquad d_p = max(0, d_b - d_r)$$
$$d_b = d_c * \sqrt{(640)^2 + (480)^2} \qquad d_r = \sqrt{(x_e - x_p)^2 + (y_e - y_p)^2}$$

where:

- x_a and y_a are the coordinates of the point the robot will aim towards.
- x_e and y_e are the coordinates of the evading robot.
- x_p and y_p are the coordinates of the pursuing robot.
- d_p is the projection distance between the evading robot and the aim point. It is set to zero if the distance between the two robots exceeds the base distance.
- d_b is the base distance; the algorithm will set the aim point such that the sum between the distance from E to the aim point plus the distance from E to P are equal to this base distance. If the distance between the two robots is already higher than the base distance, the aim position is set to the robot's current location, stopping it until the other robot is close enough.
- d_c is the base distance coefficient, given by the *dist_coef* parameter. It multiplies the maximum distance in the workspace (Euclidean distance between opposite corners) to yield the base distance.
- d_r is the Euclidean distance between the two robots.

Parameters

- *dist_coef*: Base distance coefficient.

Perpendicular to pursuer

E moves perpendicularly to the line connecting E and P, moving faster the closer the robots are. The distance that the aim point is projected away from E is calculated in the same way as in the previous strategy, but the vector between the two is rotated 90º counter-clockwise:

$$x_a = x_e - (\frac{|y_e - y_p|}{d_r}) * d_p \qquad y_a = y_e + (\frac{|x_e - x_p|}{d_r}) * d_p \qquad d_p = max(0, d_b - d_r)$$
$$d_b = d_c * \sqrt{(640)^2 + (480)^2} \qquad d_r = \sqrt{(x_e - x_p)^2 + (y_e - y_p)^2}$$

where:

- All named variables are the same as in the previous set of equations.
- When considering the vector between E and the aim point from the previous strategy as $(dx, dy) = ((\frac{|x_e - x_p|}{d_r}) * d_p, (\frac{|y_e - y_p|}{d_r}) * d_p)$, we can use it directly for this strategy by rotating it: the vector between E and the aim point in this strategy is $(-dy, dx)$.

Parameters

- *dist_coef*: Base distance coefficient.

Circular

E moves in a large, uninterrupted circle around the whole workspace, independently of P's position. The aim point is calculated in the following manner:

$$\widehat{a} = arctan(\frac{y_r - y_c}{x_r - x_c}) + \frac{\pi}{4} \qquad x_a = x_c + cos(\widehat{a}) * r * \sqrt{2} \qquad y_a = y_c - sin(\widehat{a}) * r * \sqrt{2}$$

where:

- x_c and y_c are the coordinates of the center of the circle that the robot will follow, set here at the center of the workspace: $x_c = (\frac{640}{2})$; $y_c = (\frac{480}{2})$
- x_r and y_r are the coordinates of the evading robot.
- $\widehat{a}$ is the angle at which the aim point will be projected from the center of the circle.
- x_a and y_a are, respectively, the x and y values of the point the robot will aim towards. This point will be placed on the tangent of the circle with center (x_c, y_c) and radius r at the point where the segment between the center of the circle and the position of the evading robot intersects the circle, at a distance r from the robot, in the counter-clockwise direction.
- r is the radius of the circle that the robot will describe. We set this radius to 45% of the height of the workspace (480 pixels), in order to have the robot follow a wide trajectory while limiting the risk of the evading robot escaping the workspace.

6 Battery testing

6.1 Test methodology

Exhaustive testing was conducted in order to test every combination of the pursuit and evasion strategies, and each of the parameters for the strategies tested for multiple values within pre-determined ranges. In total, 403 such combinations of strategies and parameters were tested, each combination iterated 10 times, for a total of 4030 individual tests.

6.1.1 Testing algorithm

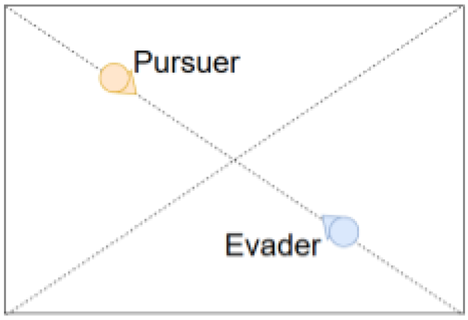

Figure 6.1: Position of robots at test start

1. Each one of the robots takes its starting position. For the pursuing robot, the starting position is the midpoint between the center of the workspace and the upper-left corner; for the evading robot, the starting position is the midpoint between the center of the the workspace and the lower-right corner. Both of the robots turn to face each other's starting points. This initial position is shown on figure 6.1.
2. The tests starts, and each of the robots moves in accordance to its corresponding strategy. There are two possible outcomes:
    a. The evading robot is captured.
    b. 50 seconds elapse without a capture taking place. This is counted as the evading robot having managed to escape capture successfully.
3. The robots exchange their roles, the evader in this test becoming the pursuer in the next test and vice versa, before moving to their starting position for the next test.

6.1.2 Tracked variables

- Time to capture (50 seconds if capture is not achieved): mean and standard deviation over 10 iterations of the test with identical strategy choice and parameters.
- Mean distance between the robots.
- Mean positional error for P (distance from P to the point it sets as its aim point).

6.1.3 Parameter values tested

Below is a list of the parameters that were introduced in the test: name of the parameter, strategies it applies to, and the values tested for that parameter. Every strategy was tested against every possible scenario, regardless of whether any of the two strategies take a parameter.

Pursuit strategies

- ***dist*** *(strats. 2, 3)*: [20, 40, 60, 80, 100, 120, 140]
- ***coef_func*** *(strat. 3)*: [1 (Linear), 2 (Square root), 3 (Quadratic)]

Evasion strategies

- ***dist_coef*** *(strats. 2, 3)*: [0.2, 0.3, 0.4, 0.5, 0.6, 0.7]

6.2 Strategy comparison

For simplicity and for greater ease of analysis, pursuit strategy 1 (direct chase) is considered a variant of strategy 2 (interception) with *dist* equal to 0, since they are functionally equivalent (projecting a point ahead of the robot's location to a distance of 0 pixels has the same result as simply putting the aim point at the robot's location).

Depending on the combination of pursuit and evasion strategies, either one or two parameters will take part in the scenario. In cases where only one parameter is involved, the results are presented in the form of a table, showing the recorded mean times to capture and the standard deviations of each of the test sets. In cases where two parameters are involved, the results are separated into two graphs, each representing the data in relation to one of the parameters. All mean times resulting from using that parameter value are averaged together to get a mean value, and the highest and lowest value from that set are displayed on the graph as well. In this manner, we can ensure that both representations offer a similar amount and type of information.

On tables, mean capture times higher than 25 seconds have been marked in green; standard deviation values higher than 1 have been marked in orange and values higher than 5 have been marked in red.

6.2.1 Direct chase / interception pursuit against blind evasion

As illustrated by the values in table 6.1, this combination of strategies tends to deal the same result and trajectory through executions: the evading robot moves away from the pursuer, soon reaching the corner, and then stays trapped against the corner until it is reached by the pursuer. There is, however, an edge case worth mentioning. When at the corner, having overlapped exactly with its aim point, the evading robot doesn't move. Due to the noise introduced by the low-resolution camera, however, the tracked position of the robot's color markers slightly fluctuates, leading the evading robot to modify its position slightly to move towards the programmatically still aim point. This can cause it to rotate and face the pursuer, despite staying on the same spot. Since the pursuer always moves towards the point it projects in front of the robot (based on the evader's orientation) and ignores the robot's actual position, at higher projection distances and with the evader facing towards the pursuer, the point is projected between them, outside of the capture range. This leads to the pursuer stopping at the projected point, without getting close enough to the evading robot to enter the capture range.

Since one of the conditions for this behavior to present itself (the evading robot's orientation) is dictated by the camera noise, different executions lead to wildly different capture times depending on a random variable, explaining the higher capture time mean and the dramatically high standard deviation.

Parameters	Time to capture	
dist (pursuit)	Mean	Std. deviation
0	11.565849	0.408677
20	11.640766	0.381351
40	11.874961	0.881832
60	11.576473	0.704307
80	11.525309	0.784344
100	13.26563	1.874453
120	25.541314	28.79118
140	27.529153	14.380772

Table 6.1: Results of pursuit strategies 1 and 2 against evasion strategy 1

Parameters	Time to capture	
dist (pursuit)	Mean	Std. deviation
0	50.0	0.0
20	46.913416	9.058366
40	50.0	0.0
60	40.324883	14.691928
80	10.771286	2.44483
100	8.764007	0.515168
120	8.003865	0.316755
140	7.773188	0.357879

Table 6.2: Results of pursuit strategies 1 and 2 against evasion strategy 4

6.2.2 Direct chase / interception pursuit against naïve evasion

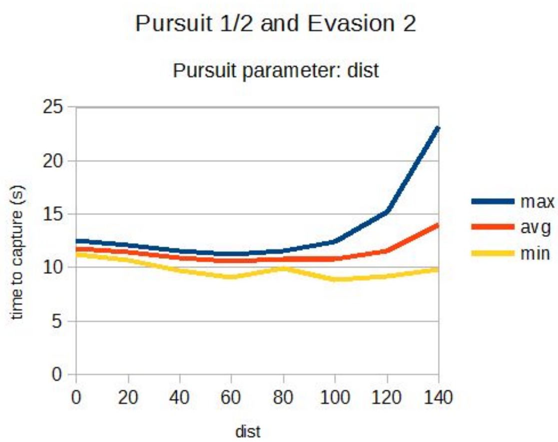

Figure 6.2: Results of pursuit strategies 1 and 2 against evasion strategy 2, respective to pursuit strategy parameter *dist*

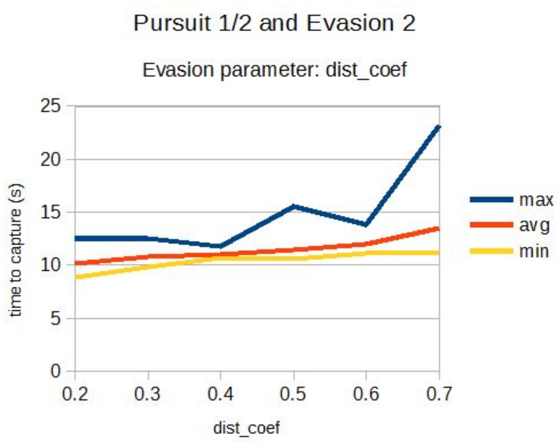

Figure 6.3: Results of pursuit strategies 1 and 2 against evasion strategy 2, respective to evasion strategy parameter *dist_coef*

This evasion tactic results in a behavior similar to the previous scenario: the evading robot moves directly to the corner, then waits to be captured. On figure 6.2 we see that the *dist* parameter follows the same pattern as described in the previous section: a much wider range of capture times in same-parameter runs when the *dist* parameter is high enough, caused by the pursuer stopping at the aim point rather than attempting to approach the evading robot. We also see on figure 6.3 that the evader manages to escape for a longer period of time as the distance coefficient rises. When the distance coefficient is low, the evading robot only starts moving when it's almost in capture range, leading to very low and consistent capture times.

6.2.3 Direct chase / interception pursuit against perpendicular-to-pursuer evasion

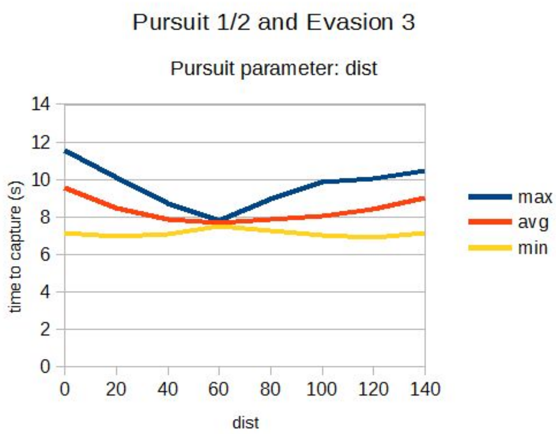

Figure 6.4: Results of pursuit strategies 1 and 2 against evasion strategy 3, respective to pursuit parameter *dist*

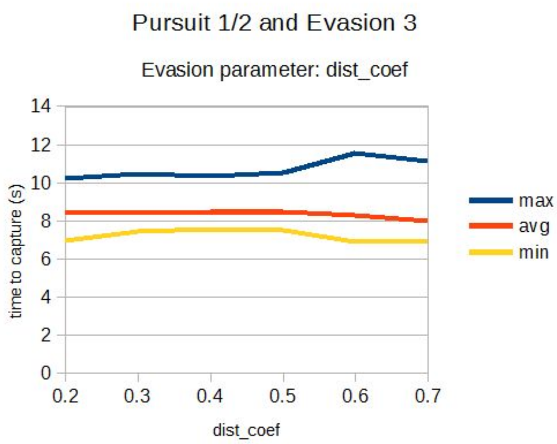

Figure 6.5: Results of pursuit strategies 1 and 2 against evasion strategy 3, respective to evasion parameter *dist_coef*

An interesting pattern appears in this scenario: the capture time is strongly tied to the distance to which the aim point is projected in front of the evading robot, reaching extremely consistent times and a local minimum when the distance approaches 60 pixels (figure 6.4). On figure 6.5 we observe that the capture time is largely independent of the distance coefficient when it is low, leading to very similar results. However, we start seeing a change in the values as the coefficient reaches 0.6 and 0.7; it is possible that these coefficients are simply too small to be significant and that more significant patterns would emerge if the coefficient parameter was high enough to lead the robot to begin moving earlier.

6.2.4 Direct chase / interception pursuit against circular evasion

In the results on table 6.2 we observe that, at low projection distances, the pursuer only manages to capture the evader in very few cases, slightly lowering the mean capture time but significantly raising the standard deviation. However, as the projection distance rises, the pursuing robot is able to speed up to catch up to the evader by keeping a higher speed and moving to its future predicted position instead of lagging behind as it escapes in its circular trajectory.

6.2.5 Dynamic interception pursuit against blind evasion

On figure 6.6, we see a clear downward trend as the *dist* parameter grows. This can be attributed to the way the pursuing robot moves depending on its distance to the aim point, as the robots are programmed to slow down when close to their aim point in order not to surpass it. Because the evading robot's first motion is to turn away and go towards the corner, the aim point for the pursuing robot is set further away, speeding it up. As the pursuer's aim point moves towards the evading robot as the pursuer approaches the evader, the problem described in 6.2.1 does not present itself, allowing the pursuer to reach the evader even if the latter is facing towards the former.

When comparing the three coefficient functions against each other, we can see that the square root function (which causes the distance between the pursuer's aim point and the evading robot to vary faster when the two robots are closer) surpasses both of the others comfortably. This can be attributed to the fact that the aim point stays at a greater distance for a longer period of time, thus enabling the pursuer to maintain a higher speed over most of its trajectory.

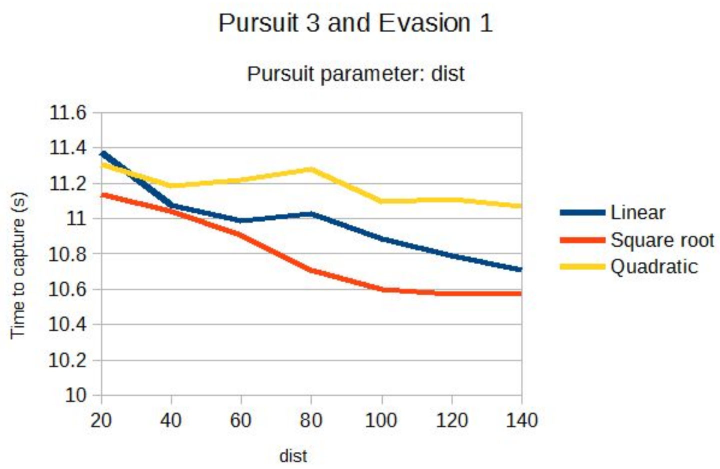

Figure 6.6: Results of pursuit 3 against evasion 1, respective to pursuit parameter *dist*

6.2.6 Dynamic interception pursuit against naïve evasion

Linear

The downward trend described in 6.2.5 is also observed in figure 6.7.

Figure 6.8 can be read as encompassing three regions:

- Between 0.2 and 0.3, the capture time ascends. The capture time for 0.2 is low because the evading robot only starts moving when the pursuing robot is almost within capture range. On the other hand, the time for 0.3 is high: this is attributed to the evading robot moving slowly away from the pursuer while staying just outside of the capture distance; because the two robots are close together, the pursuing robot moves slowly towards its aim point. This combination of slow movement means that both of the robots move slowly towards the corner, thus taking longer to reach it and extending the capture time.

- Between 0.3 and 0.5, the evading robot speeds up towards the corner, making each trip slightly shorter, and reducing the time it takes to get to the corner and thus to make the capture.
- Between 0.5 and 0.7, the capture time rises again. This is attributed to the evading robot being able to reach the corner earlier and turn around, putting the pursuer's aim point closer to the pursuer and thus slowing it down. This is also evidenced by the growing standard deviation value (a complete table of results including standard deviation for each scenario is provided in Appendix A), since the orientation that the robot adopts once it has reached its destination is dependent on the random noise appearing on the image, as explained in 6.2.1.

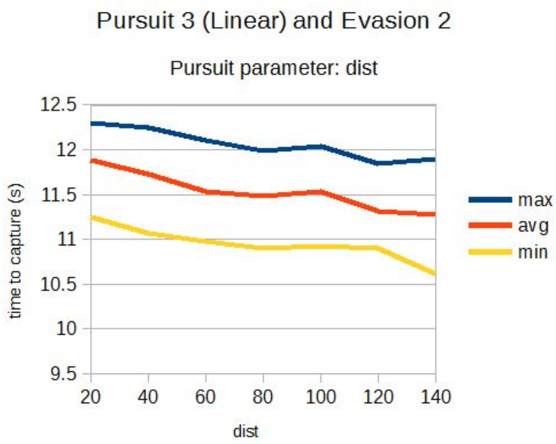

Figure 6.7: Results of pursuit strategy 3 with linear variation against evasion strategy 2, respective to pursuit parameter *dist*

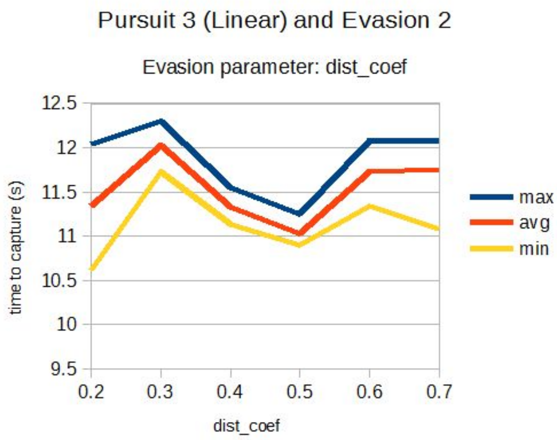

Figure 6.8: Results of pursuit strategy 3 with linear variation against evasion strategy 2, respective to evasion parameter *dist_coef*

Square root

The downward trend described in 6.2.5 is also observed in figure 6.9.

The distance coefficient parameter displayed on figure 6.10 affects the results in a manner similar to the one explained in the linear case; with the difference that, due to the nature of the square root coefficient function (explained in 6.2.5), the pursuer's aim point stays far away enough in front of the evader for the pursuer to always manage to catch up to the evader earlier (because, since its aim point is further away, the pursuer maintains top speed for a longer period of time).

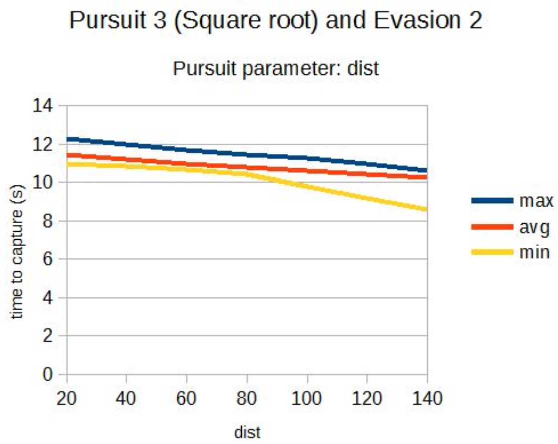

Figure 6.9: Results of pursuit strategy 3 with square root variation against evasion strategy 2, respective to pursuit parameter *dist*

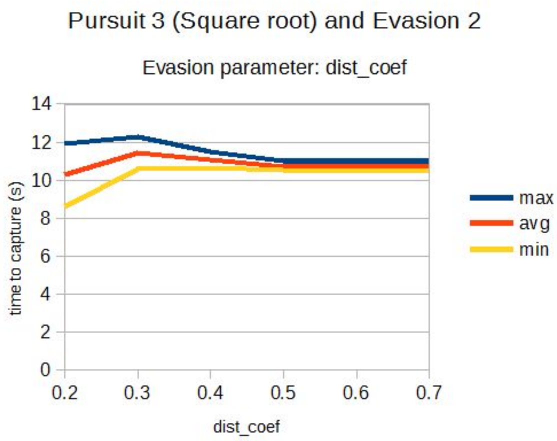

Figure 6.10: Results of pursuit strategy 3 with square root variation against evasion strategy 2, respective to evasion parameter *dist_coef*

Quadratic

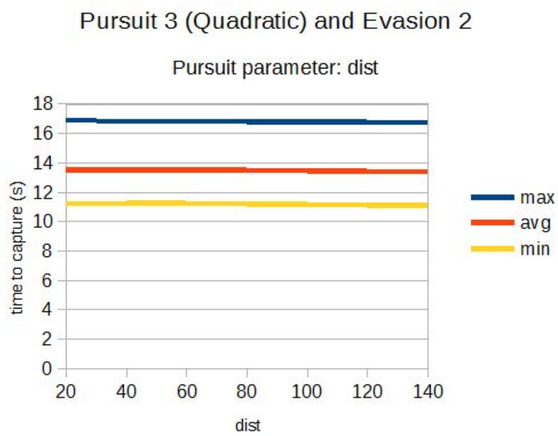

Figure 6.11: Results of pursuit strategy 3 with quadratic variation against evasion strategy 2, respective to pursuit parameter *dist*

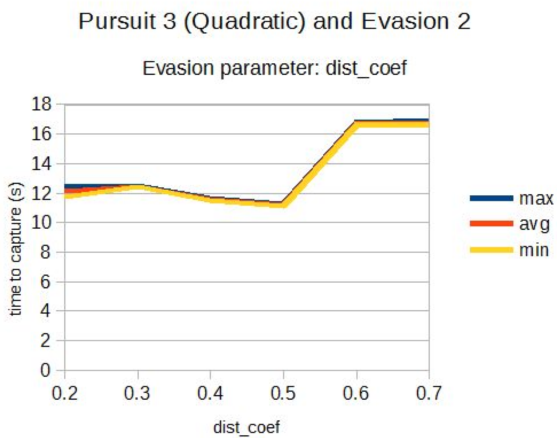

Figure 6.12: Results of pursuit strategy 3 with quadratic variation against evasion strategy 2, respective to evasion parameter *dist_coef*

We see on figure 6.11 that the capture time appears to be largely independent of the *dist* parameter. The quadratic coefficient function causes the distance between the pursuer's aim point and the evading robot to vary faster when the two robots are further apart. Taking this into account, it is possible that, by the time the distance between the pursuing robot and its aim point is small enough to cause the robot to move at less than maximum speed, the aim point is already too close to the evading robot for it to lead to any significant change in speed.

Once again, the distance coefficient graph (figure 6.12) shows a similar shape to the one it shows in the other two cases in this section. Because the quadratic coefficient function moves the pursuer's aim point towards the evader sooner (when the robots are far apart), it is much more consistent than the other two when faced with an evading robot that turns around on the spot (and thus moves the aim point away from itself). There are two rather than three areas in this graph: before 0.5, it is probable that the robot was captured before it had the time to start moving. After 0.6, the robot starts moving earlier, and manages to make it to the corner before being captured, hence the difference in times.

6.2.7 Dynamic interception against perpendicular-to-pursuer evasion

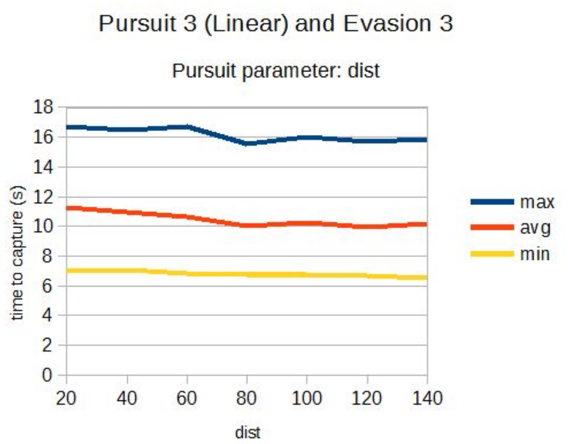

Figure 6.13: Results of pursuit strategy 3 with linear variation against evasion strategy 3, respective to pursuit parameter *dist*

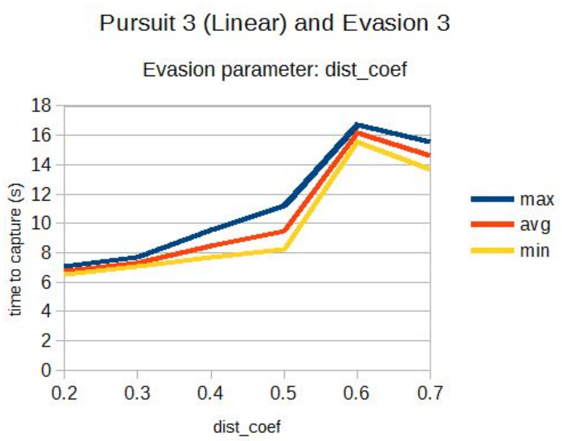

Figure 6.14: Results of pursuit strategy 3 with linear variation against evasion strategy 3, respective to evasion parameter *dist_coef*

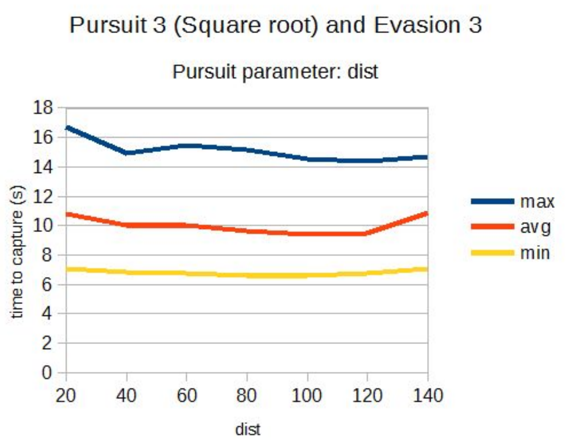

Figure 6.15: Results of pursuit strategy 3 with square root variation against evasion strategy 3, respective to pursuit parameter *dist*

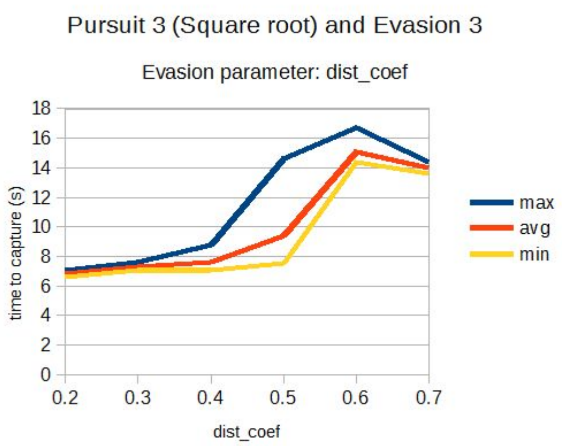

Figure 6.16: Results of pursuit strategy 3 with square root variation against evasion strategy 3, respective to evasion parameter *dist_coef*

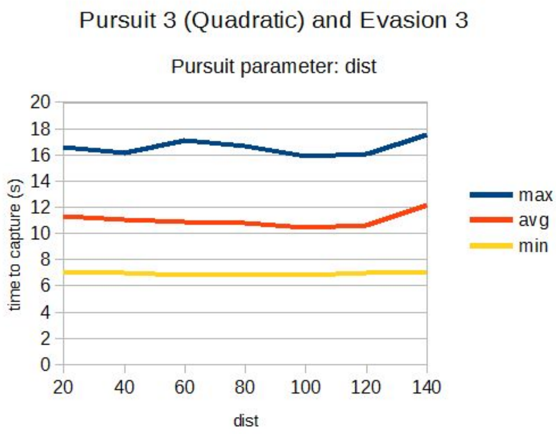

Figure 6.17: Results of pursuit strategy 3 with quadratic variation against evasion strategy 3, respective to pursuit parameter *dist*

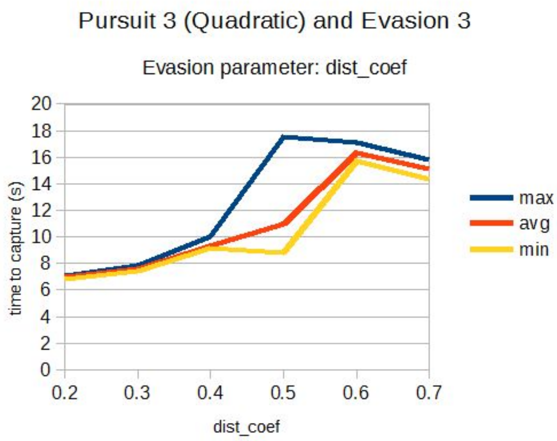

Figure 6.18: Results of pursuit strategy 3 with quadratic variation against evasion strategy 3, respective to evasion parameter *dist_coef*

All three of the coefficient functions behave in a similar way in this scenario. As shown on the graphs respective to the *dist* parameter (figures 6.13, 6.15 and 6.17), the time to capture appears to be largely independent of the projection distance parameter. However, the distance coefficient graphs (figures 6.14, 6.16 and 6.18) vary greatly:

- At lower *dist_coef* values, the evading robot only starts moving when the pursuing robot is already too close to reaching the capture zone, and thus is quickly captured.
- As the value of the parameter rises, the evading robot starts its movement earlier, forcing the pursuing robot to change its trajectory and avoiding capture for a longer period of time.
- At 0.7, the evading robot's aim point is projected at a farther distance. This distance is far enough to surpass the upper threshold value in its movement strategy (strategy 3, described in 3.1.3), and the robot stops moving forward in order to turn towards its aim point. That stop in the evading robot's movement grants the pursuing robot enough time to catch up to it and capture it.

6.2.8 Dynamic interception against circular evasion

Every test in this category yielded the same result: the evading robot successfully managed to escape capture for the duration of the 50-second time limit. This behavior can be compared to the one observed in 6.2.4 at low *dist* values; even with the higher base *dist* values, the reduction of the distance applied by the reduction coefficient is significant enough to bring them into the observed no-capture area again. It is possible that, if higher base *dist* values were set, this behavior would vary.

6.3 Analysis conclusions

Some recognizable patterns emerged from the testing:

- The blind and naïve evasion strategies always produce a nearly identical evader robot trajectory: the evading robot moves directly away from the pursuing robot, stopping upon reaching the corner of the field and staying there until the pursuer reaches it.
- With low *dist_coef* values, the evading robot starts moving too late to avoid capture close to its starting point.
- The perpendicular-to-pursuer evasion strategy took the most advantage of the robots' maneuvering restraints by forcing the pursuer to turn on the spot. If the movement of circling around the pursuer was modified to also allow the evader to move away from the pursuer, this new evasion strategy might achieve better results.
- The circular evasion strategy was the only strategy that reliably escaped capture, showing results consistent with the solutions proposed for the Lion and man problem in continuous time in [12] and [13].
- Interception pursuit strategies tend to be more effective the larger the projection distance is, as shown by the downward trend of the data as the *dist* parameter grows.

7 Conclusions and future work

This project improves and expands a software and hardware platform for visual servoing, enhancing the precision of the trajectories described by the robots it controls and enabling it to control multiple robots with different objectives simultaneously, determining their trajectories and their movement instructions in real time. Furthermore, a practical application of this system is demonstrated, in testing simple movement strategies for a single-evader, single-pursuer, continuous bounded space version of the pursuit-evasion problem. This determines the validity of the system and shows its usefulness in real-world applications.

With this work, we show that, with the low cost of the components and simple setup of the system, it is possible to conduct exhaustive experiments on a physical representation of a mathematical problem, providing a first empirical base for a problem from which to draw estimates and patterns without the need for a more complex and demanding software simulation.

7.1 Future work

The next steps on this project could be taken in several different directions, depending on the final applications that the platform will be modified to address.

Firstly, a comparative research regarding a replacement for the camera could be conducted. If this step is taken first, it opens the way for methods of robot recognition not based on color, such as pattern recognition (QR codes). With the current image capture quality, it is not possible to capture such a pattern with a resolution high enough for the pattern to be recognizable. After this modification is implemented and validated, more robots could be added to the system, allowing the testing of other problems with more than two players.

Another available path can be undertaken by improving the robots’ control algorithm, making the robots more accurate and quicker. With the implementation of a learning algorithm such as a neural network, the dependency on human-designed algorithms would be overcome, and a more optimal control scheme would emerge from repeated testing.

Finally, a third option would be to expand the system by adding more cameras, enabling the system to encompass a larger workspace. For this to be viable, the software platform would be modified to keep track of the relative position of the cameras along the worskpace, allowing an image of the complete workspace from different perspectives to be “stitched” together, collecting environmental information from a larger area.

Bibliography

[1] Márquez Pardo, Iván. “Visual control of a mobile robot by means of an overhead view.” BS thesis. 2016. URL: https://repositorio.uam.es/handle/10486/676760

[2] Isler, Volkan, Geoffrey A. Hollinger, and Timothy H. Chung. “Search and Pursuit-Evasion in Mobile Robotics, A survey.” (2011). URL: http://www-users.cs.umn.edu/~isler/pub/pe-survey.pdf

[3] Karnad, Nikhil. “The Role of Sensing Information in Pursuit-Evasion Games.” Diss. Rensselaer Polytechnic Institute, 2008. URL: https://www.cs.rpi.edu/twiki/pub/RoboticsWeb/LabPublications/Kms08.pdf

[4] Klein, Kyle, and Subhash Suri. “Complete Information Pursuit Evasion in Polygonal Environments.” AAAI. 2011. URL: https://www.cs.ucsb.edu/~suri/psdir/aaai11.pdf

[5] Shedied, Samy Aly. “Optimal Control for a Two Player Dynamic Pursuit Evasion Game; The Herding Problem.” Diss. 2002. URL: https://theses.lib.vt.edu/theses/available/etd-02042002-152330/unrestricted/Dissertation.pdf

[6] Jang, Jung Soon, and Claire J. Tomlin. “Control strategies in multi-player pursuit and evasion game.” AIAA 6239 (2005): 15-18. URL: https://people.eecs.berkeley.edu/~tomlin/papers/conferences/jt05_gnc.pdf

[7] Borie, Richard B., Craig A. Tovey, and Sven Koenig. “Algorithms and Complexity Results for Pursuit-Evasion Problems.” IJCAI. Vol. 9. 2009. URL: https://www.aaai.org/ocs/index.php/IJCAI/IJCAI-09/paper/viewFile/482/591

[8] Vieira, Marcos AM, Ramesh Govindan, and Gaurav S. Sukhatme. “Optimal policy in discrete pursuit-evasion games.” Department of Computer Science, University of Southern California, Tech. Rep (2008): 08-900. URL: https://www.cs.usc.edu/assets/001/82869.pdf

[9] Vieira, Marcos AM, Ramesh Govindan, and Gaurav S. Sukhatme. “Scalable and practical pursuit-evasion.” Robot Communication and Coordination, 2009. ROBOCOMM'09. Second International Conference on. IEEE, 2009. URL: https://www.cs.usc.edu/assets/001/82871.pdf

[10] Droidcam project, Dev47Apps, 2010-2016, URL: https://www.dev47apps.com/

[11] Android Debug Bridge, URL: https://developer.android.com/studio/command-line/adb.html

[12] J. E. Littlewood, “A Mathematician’s Miscellany.” 1953, Methuen & Co. Ltd., London

[13] Sgall, Jiří. “Solution of David Gale's lion and man problem.” Theoretical Computer Science 259.1 (2001): 663-670.

[14] Hutchinson, Seth, Gregory D. Hager, and Peter I. Corke. “A tutorial on visual servo control.” IEEE transactions on robotics and automation 12.5 (1996): 651-670. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=538972

[15] Shaw, J., and K. Y. Cheng. “Object identification and 3-D position calculation using eye-in-hand single camera for robot gripper.” Industrial Technology (ICIT), 2016 IEEE International Conference on. IEEE, 2016. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7475004

[16] Léonard, Simon, Elizabeth A. Croft, and James J. Little. “Planning collision-free and occlusion-free paths for industrial manipulators with eye-to-hand configuration.” Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. IEEE, 2009.

[17] Muis, Abdul, and Kouhei Ohnishi. “Eye-to-hand approach on eye-in-hand configuration within real-time visual servoing.” IEEE/ASME transactions on Mechatronics 10.4 (2005): 404-410.

[18] Luo, Ren C., et al. “Hybrid Eye-to-hand and Eye-in-hand visual servo system for parallel robot conveyor object tracking and fetching.” Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE. IEEE, 2014.

[19] Merz, A. W. “The homicidal chauffeur.” AIAA Journal 12.3 (1974): 259-260.

[20] Aigner, Martin, and Michael Fromme. “A game of cops and robbers.” Discrete Applied Mathematics 8.1 (1984): 1-12.

[21] Samet, Hanan. “The design and analysis of spatial data structures.” Vol. 199. Reading, MA: Addison-Wesley, 1990.

[22] Krupa, Alexandre, et al. “Autonomous 3-D positioning of surgical instruments in robotized laparoscopic surgery using visual servoing.” IEEE transactions on robotics and automation 19.5 (2003): 842-853.

[23] Rizzi, Alfred A., Jay Gowdy, and Ralph L. Hollis. “Distributed coordination in modular precision assembly systems.” The International Journal of Robotics Research 20.10 (2001): 819-838.

[24] Mejias, Luis, et al. “Visual servoing of an autonomous helicopter in urban areas using feature tracking.” Journal of Field Robotics 23.3-4 (2006): 185-199.

[25] Chaumette, François, and Seth Hutchinson. “Visual servo control. I. Basic approaches.” IEEE Robotics & Automation Magazine 13.4 (2006): 82-90.

[26] Pieters, Roel. “Visual Servo Control.” Eindhoven University of Technology, The Netherlands (2012).

[27] Trucco, Emanuele, and Konstantinos Plakas. “Video tracking: a concise survey.” IEEE Journal of Oceanic Engineering 31.2 (2006): 520-529.

[28] Yousf, Wael Mohamed, Osama Mohamed Elmowafy, and Ibrahim Ali Abdl-Dayem. “C18. Modified CAMShift algorithm for adaptive window tracking.” Radio Science Conference (NRSC), 2012 29th National. IEEE, 2012.

Appendices

Appendix A: Complete battery testing results

Pursuit strat	Evasion strat	Pursuit *coef_func*	Pursuit *dist*	Evasion *dist_coef*	Time to capture: mean	Time to capture: std. dev.	Mean positional error	Mean distance to target
1	1	0	0	0	11.565849	0.408677	222.354818	222.354818
1	2	0	0	0.2	12.534734	0.722	171.110581	171.110581
1	2	0	0	0.3	12.543946	0.115503	188.842294	188.842294
1	2	0	0	0.4	11.797313	0.292805	211.072233	211.072233
1	2	0	0	0.5	11.347157	0.374261	226.328388	226.328388
1	2	0	0	0.6	11.253179	0.345023	225.790371	225.790371
1	2	0	0	0.7	11.366871	0.416348	223.83925	223.83925
1	3	0	0	0.2	7.15177	0.297559	229.051399	229.051399
1	3	0	0	0.3	7.65737	0.293238	221.612054	221.612054
1	3	0	0	0.4	9.333097	1.691399	196.217639	196.217639
1	3	0	0	0.5	10.496157	2.137057	186.758472	186.758472
1	3	0	0	0.6	11.524499	1.547683	180.080574	180.080574
1	3	0	0	0.7	11.109107	1.022322	182.266958	182.266958
1	4	0	0	0	50.0	0.0	166.343423	166.343423
2	1	0	20	0	11.640766	0.381351	237.543268	224.944103
2	1	0	40	0	11.874961	0.881832	256.216386	234.044152
2	1	0	60	0	11.576473	0.704307	262.553907	230.717328
2	1	0	80	0	11.525309	0.784344	269.095517	229.601708
2	1	0	100	0	13.26563	1.874453	287.337573	247.881632
2	1	0	120	0	25.541314	28.79118	169.45034	190.427662
2	1	0	140	0	27.529153	14.380772	179.822004	196.743894
2	2	0	20	0.2	10.710384	0.904998	196.641008	181.253075
2	2	0	20	0.3	12.050085	0.108479	201.74204	186.691019

2	2	0	20	0.4	11.321266	0.124911	224.501666	210.275174
2	2	0	20	0.5	11.014874	0.323636	237.651605	224.884805
2	2	0	20	0.6	11.505941	0.506315	231.981785	219.443669
2	2	0	20	0.7	11.744282	0.892158	220.678137	208.08378
2	2	0	40	0.2	9.710183	1.344376	219.242798	189.65518
2	2	0	40	0.3	11.299255	0.165668	219.161986	189.344877
2	2	0	40	0.4	11.054851	0.108568	235.102584	208.608479
2	2	0	40	0.5	10.737707	0.307252	248.658742	225.330573
2	2	0	40	0.6	11.505843	0.703744	234.657747	210.078546
2	2	0	40	0.7	11.299011	0.549282	236.194216	212.182989
2	2	0	60	0.2	9.092376	0.826383	236.689578	193.349645
2	2	0	60	0.3	10.715944	0.301855	234.876071	191.11026
2	2	0	60	0.4	10.765066	0.130059	244.021817	205.93838
2	2	0	60	0.5	10.583777	0.310249	256.750172	223.600379
2	2	0	60	0.6	11.076456	0.50253	250.796721	217.019302
2	2	0	60	0.7	11.187188	0.81677	246.345207	213.15795
2	2	0	80	0.2	9.921321	1.757995	244.104704	186.784603
2	2	0	80	0.3	10.289476	0.392729	250.142833	193.995418
2	2	0	80	0.4	10.722474	0.196821	253.344295	205.560506
2	2	0	80	0.5	10.544165	0.3224	262.946513	221.406314
2	2	0	80	0.6	11.506435	1.158193	258.594355	219.350191
2	2	0	80	0.7	11.597218	0.764864	259.658444	219.744767
2	2	0	100	0.2	8.813171	1.441051	269.377707	198.91782
2	2	0	100	0.3	9.984016	0.557657	262.788463	195.336487
2	2	0	100	0.4	10.88632	0.565087	263.010468	206.983904
2	2	0	100	0.5	10.586014	0.292268	276.203863	227.849898
2	2	0	100	0.6	11.763294	1.132757	268.596805	222.965563
2	2	0	100	0.7	12.396461	1.329592	274.675023	234.485834
2	2	0	120	0.2	9.143617	1.534515	280.979504	198.982134

2	2	0	120	0.3	9.832963	0.51077	274.848291	196.733659
2	2	0	120	0.4	10.867331	0.403475	272.462611	209.426529
2	2	0	120	0.5	10.950836	1.137023	274.562091	223.474037
2	2	0	120	0.6	13.398986	3.294044	270.473138	226.245041
2	2	0	120	0.7	15.204981	9.50868	234.111169	212.624887
2	2	0	140	0.2	11.212492	2.690205	285.729156	197.929764
2	2	0	140	0.3	9.804403	0.465996	285.442983	198.968208
2	2	0	140	0.4	10.720971	0.285761	275.688438	205.912742
2	2	0	140	0.5	15.549438	7.434179	234.373286	194.81753
2	2	0	140	0.6	13.791543	5.980688	272.052994	225.470028
2	2	0	140	0.7	23.23601	13.290872	202.918198	182.188136
2	3	0	20	0.2	6.925306	0.214124	235.703166	227.800595
2	3	0	20	0.3	7.445424	0.207379	226.158387	221.22094
2	3	0	20	0.4	8.190706	0.822925	211.873682	209.121422
2	3	0	20	0.5	9.373589	1.082166	197.345231	196.990004
2	3	0	20	0.6	10.082922	1.550571	191.505574	190.12402
2	3	0	20	0.7	8.755916	0.80535	201.241473	201.74475
2	3	0	40	0.2	7.086295	0.400021	240.132548	223.410512
2	3	0	40	0.3	7.447621	0.29138	231.495471	220.144629
2	3	0	40	0.4	7.654012	0.384079	224.70155	218.667026
2	3	0	40	0.5	8.229085	1.087558	210.657238	211.291379
2	3	0	40	0.6	8.679757	0.819686	204.602615	203.095787
2	3	0	40	0.7	8.05504	0.481753	207.940842	209.277214
2	3	0	60	0.2	7.672919	0.16756	238.120281	212.11789
2	3	0	60	0.3	7.476109	0.313974	238.397092	219.901729
2	3	0	60	0.4	7.599411	0.525259	231.011331	220.463113
2	3	0	60	0.5	7.801407	0.532574	221.662633	222.522195
2	3	0	60	0.6	7.735724	0.561114	216.244223	216.146239
2	3	0	60	0.7	7.603243	0.449466	216.132292	218.090589

2	3	0	80	0.2	8.930613	0.856951	232.665035	195.567763
2	3	0	80	0.3	8.434648	0.517713	236.113823	209.718496
2	3	0	80	0.4	7.573107	0.550002	237.576475	222.438307
2	3	0	80	0.5	7.567542	0.383091	230.807816	231.875471
2	3	0	80	0.6	7.346562	0.367242	223.817287	225.174749
2	3	0	80	0.7	7.24216	0.309617	222.640177	225.45148
2	3	0	100	0.2	9.845637	0.220087	235.498954	187.087532
2	3	0	100	0.3	8.707484	0.438252	245.096175	210.906109
2	3	0	100	0.4	8.123943	0.835307	241.646147	220.167869
2	3	0	100	0.5	7.490269	0.210536	236.594275	235.084994
2	3	0	100	0.6	6.981882	0.090589	229.496401	233.348846
2	3	0	100	0.7	7.058549	0.24466	223.883139	228.188504
2	3	0	120	0.2	10.064812	1.233421	248.400066	186.995393
2	3	0	120	0.3	9.515757	0.299924	253.605791	208.873725
2	3	0	120	0.4	8.990381	1.253799	242.129704	218.013705
2	3	0	120	0.5	7.972094	0.41718	237.050592	238.650919
2	3	0	120	0.6	6.868769	0.253664	231.454277	237.989732
2	3	0	120	0.7	6.922415	0.383576	225.326581	232.003307
2	3	0	140	0.2	10.200571	1.368248	262.452052	186.746229
2	3	0	140	0.3	10.460845	0.464914	258.28201	205.415103
2	3	0	140	0.4	10.311556	0.877735	250.913563	225.237761
2	3	0	140	0.5	8.932187	0.780043	241.130906	241.399268
2	3	0	140	0.6	7.105546	0.431279	230.232373	237.753923
2	3	0	140	0.7	7.10328	0.728106	228.785848	236.367226
2	4	0	20	0	46.913416	9.058366	170.400417	156.686969
2	4	0	40	0	50.0	0.0	175.241315	146.392512
2	4	0	60	0	40.324883	14.691928	182.643206	139.789782
2	4	0	80	0	10.771286	2.44483	229.129819	199.445849
2	4	0	100	0	8.764007	0.515168	249.980995	219.519922

2	4	0	120	0	8.003865	0.316755	263.393852	235.645815
2	4	0	140	0	7.773188	0.357879	273.507681	239.535768
3	1	1	20	0	11.376904	0.434749	231.997523	228.016134
3	1	2	20	0	11.138022	0.441538	235.489675	227.795981
3	1	3	20	0	11.307565	0.380442	228.501557	227.479623
3	1	1	40	0	11.078686	0.351567	236.478607	228.156874
3	1	2	40	0	11.044477	0.378357	237.808221	224.061184
3	1	3	40	0	11.184357	0.372367	230.62966	228.348915
3	1	1	60	0	10.984934	0.368738	239.394147	227.270192
3	1	2	60	0	10.906434	0.427608	244.199944	224.661137
3	1	3	60	0	11.216331	0.413066	230.075103	226.820464
3	1	1	80	0	11.02911	0.40712	242.520222	226.323131
3	1	2	80	0	10.705603	0.349932	253.836383	226.338903
3	1	3	80	0	11.277673	0.472162	227.653567	223.582731
3	1	1	100	0	10.884935	0.428454	245.594786	225.650775
3	1	2	100	0	10.597476	0.419456	261.807777	227.384895
3	1	3	100	0	11.098324	0.414529	233.705883	228.05602
3	1	1	120	0	10.786763	0.375373	249.57821	224.124542
3	1	2	120	0	10.572866	0.337809	266.14091	227.104372
3	1	3	120	0	11.106769	0.399627	233.490741	226.541748
3	1	1	140	0	10.706166	0.397448	254.435354	225.158575
3	1	2	140	0	10.578958	0.484779	271.863858	226.371149
3	1	3	140	0	11.071589	0.41799	238.80262	229.585249
3	2	1	20	0.2	12.034612	1.155704	177.339986	174.66002
3	2	1	20	0.3	12.296979	0.093191	192.035254	189.010152
3	2	1	20	0.4	11.542559	0.149054	214.320162	210.849049
3	2	1	20	0.5	11.246921	0.254449	227.835411	224.247465
3	2	1	20	0.6	12.078342	0.962609	211.536048	208.48396
3	2	1	20	0.7	12.079322	0.63494	209.634885	206.539288

3	2	1	40	0.2	11.838552	1.176037	180.477014	174.938885
3	2	1	40	0.3	12.247595	0.12153	193.632442	187.469425
3	2	1	40	0.4	11.458525	0.158907	217.863068	210.49753
3	2	1	40	0.5	11.066304	0.236438	232.544219	224.71652
3	2	1	40	0.6	11.839952	0.636032	216.417927	210.065944
3	2	1	40	0.7	11.900165	0.804626	212.292041	206.658975
3	2	1	60	0.2	11.506467	1.107869	185.115255	176.435714
3	2	1	60	0.3	12.109938	0.083851	198.011835	188.347292
3	2	1	60	0.4	11.367212	0.113928	219.804921	208.555237
3	2	1	60	0.5	10.977979	0.249992	237.56913	225.643699
3	2	1	60	0.6	11.491632	0.59795	222.970524	212.29942
3	2	1	60	0.7	11.737469	0.589091	216.728407	207.557053
3	2	1	80	0.2	11.431293	1.08233	187.527394	175.78205
3	2	1	80	0.3	11.984634	0.138843	199.98782	186.823789
3	2	1	80	0.4	11.282674	0.203805	224.911742	209.707361
3	2	1	80	0.5	10.900813	0.303659	240.797766	224.791206
3	2	1	80	0.6	11.339778	0.56992	228.128116	213.789962
3	2	1	80	0.7	11.918826	0.744714	217.980615	205.124172
3	2	1	100	0.2	11.079911	1.04009	192.962967	177.992501
3	2	1	100	0.3	11.961066	0.092702	203.937441	186.921034
3	2	1	100	0.4	11.319868	0.158599	227.73112	208.21373
3	2	1	100	0.5	10.928651	0.470787	246.55567	226.185441
3	2	1	100	0.6	12.044965	0.882619	224.416868	207.341062
3	2	1	100	0.7	11.874014	0.939245	226.89927	210.648254
3	2	1	120	0.2	10.903725	0.927303	198.008359	178.900356
3	2	1	120	0.3	11.850534	0.114515	206.78052	185.89967
3	2	1	120	0.4	11.131451	0.263104	233.229696	209.026527
3	2	1	120	0.5	10.92679	0.395356	245.166753	221.174292
3	2	1	120	0.6	11.425301	0.719904	236.683628	214.934229

3	2	1	120	0.7	11.659966	0.864114	232.915301	212.192204
3	2	1	140	0.2	10.614377	0.904536	203.305596	180.290626
3	2	1	140	0.3	11.725739	0.145784	212.156051	186.951196
3	2	1	140	0.4	11.211165	0.66844	239.28637	210.40456
3	2	1	140	0.5	11.11876	0.919672	251.159294	222.97649
3	2	1	140	0.6	11.898918	1.035565	239.374075	214.618084
3	2	1	140	0.7	11.077004	0.684782	241.407527	215.604879
3	2	2	20	0.2	11.900024	2.359256	183.374871	177.049071
3	2	2	20	0.3	12.276303	0.232399	195.551858	188.496639
3	2	2	20	0.4	11.486402	0.222403	219.560195	211.824012
3	2	2	20	0.5	11.013457	0.25176	238.437379	230.331914
3	2	2	20	0.6	11.003961	0.349044	234.815748	227.170008
3	2	2	20	0.7	10.991439	0.326043	234.554433	226.841091
3	2	2	40	0.2	11.323415	2.300604	190.755644	177.558966
3	2	2	40	0.3	11.992585	0.24847	202.448729	188.139786
3	2	2	40	0.4	11.248138	0.177567	225.693096	210.691294
3	2	2	40	0.5	10.846719	0.308047	242.961665	228.27417
3	2	2	40	0.6	10.874369	0.28771	243.696666	228.765084
3	2	2	40	0.7	10.860742	0.3468	242.473189	228.069675
3	2	2	60	0.2	10.901929	1.916735	199.63904	179.222963
3	2	2	60	0.3	11.672057	0.301836	209.351447	187.432108
3	2	2	60	0.4	11.139757	0.145355	230.511819	208.316665
3	2	2	60	0.5	10.684335	0.250737	249.701972	228.053787
3	2	2	60	0.6	10.727283	0.311607	247.494494	226.005628
3	2	2	60	0.7	10.648809	0.216628	248.627994	227.251628
3	2	2	80	0.2	10.425793	1.646536	210.809582	182.556766
3	2	2	80	0.3	11.441518	0.34779	218.783645	189.021024
3	2	2	80	0.4	11.014973	0.227276	238.780606	208.857757
3	2	2	80	0.5	10.584962	0.237975	256.480384	228.222915

3	2	2	80	0.6	10.598798	0.308104	254.255524	226.358049
3	2	2	80	0.7	10.636865	0.361385	253.955564	226.594589
3	2	2	100	0.2	9.761429	1.542597	224.426739	188.230908
3	2	2	100	0.3	11.243109	0.4544	227.159167	189.14087
3	2	2	100	0.4	11.018319	0.651059	247.300655	209.892329
3	2	2	100	0.5	10.517532	0.257487	260.990722	226.540137
3	2	2	100	0.6	10.62114	0.277406	259.924291	225.784485
3	2	2	100	0.7	10.629721	0.288492	253.838271	222.129018
3	2	2	120	0.2	9.199014	0.946648	236.012177	191.19345
3	2	2	120	0.3	10.935663	0.425978	236.07491	189.551702
3	2	2	120	0.4	10.770346	0.193899	253.080772	208.270582
3	2	2	120	0.5	10.583209	0.423206	269.873601	228.941712
3	2	2	120	0.6	10.553616	0.414424	264.404832	224.766357
3	2	2	120	0.7	10.52096	0.392975	265.852059	226.652709
3	2	2	140	0.2	8.590857	0.840813	255.182853	200.982691
3	2	2	140	0.3	10.552221	0.586244	247.093526	191.610554
3	2	2	140	0.4	10.622477	0.214433	258.308806	206.306077
3	2	2	140	0.5	10.588686	0.372586	275.873538	229.425464
3	2	2	140	0.6	10.448214	0.274534	271.46418	225.692979
3	2	2	140	0.7	10.553366	0.436569	274.227711	228.442419
3	2	3	20	0.2	12.399217	1.227049	173.442275	172.94483
3	2	3	20	0.3	12.490866	0.180283	189.483519	188.84436
3	2	3	20	0.4	11.670151	0.161037	212.042862	211.152987
3	2	3	20	0.5	11.187863	0.305374	230.616046	229.548108
3	2	3	20	0.6	16.641442	0.702661	272.577284	273.767867
3	2	3	20	0.7	16.925626	0.726166	265.995408	267.243152
3	2	3	40	0.2	12.207014	1.164178	174.620602	173.569429
3	2	3	40	0.3	12.533339	0.175097	189.874841	188.557724
3	2	3	40	0.4	11.625402	0.197137	213.133724	211.30707

3	2	3	40	0.5	11.251258	0.283878	228.949257	226.865111
3	2	3	40	0.6	16.776696	0.579086	270.267505	272.287294
3	2	3	40	0.7	16.644617	0.765008	271.399796	273.495688
3	2	3	60	0.2	12.269073	1.189099	174.203287	172.578303
3	2	3	60	0.3	12.465086	0.144357	190.561227	188.522816
3	2	3	60	0.4	11.653438	0.145329	213.331451	210.555099
3	2	3	60	0.5	11.247442	0.338459	228.983489	225.813987
3	2	3	60	0.6	16.659983	0.686045	266.99153	270.297689
3	2	3	60	0.7	16.845654	0.813337	263.865997	266.924122
3	2	3	80	0.2	12.255203	1.309373	175.508154	173.222919
3	2	3	80	0.3	12.459696	0.126106	190.806518	187.965962
3	2	3	80	0.4	11.613238	0.235551	214.753786	210.798121
3	2	3	80	0.5	11.195215	0.246446	229.920367	225.498996
3	2	3	80	0.6	16.818561	0.57704	266.759123	270.501197
3	2	3	80	0.7	16.672661	0.722527	267.944042	271.496743
3	2	3	100	0.2	12.045033	1.24212	176.484765	173.614226
3	2	3	100	0.3	12.40534	0.199113	192.391692	188.687623
3	2	3	100	0.4	11.606569	0.243719	215.9143	210.843034
3	2	3	100	0.5	11.172063	0.390945	234.506084	228.447673
3	2	3	100	0.6	16.810837	0.666379	266.32275	270.641659
3	2	3	100	0.7	16.763614	0.693182	265.981501	270.559513
3	2	3	120	0.2	11.997123	1.393753	177.072348	173.42036
3	2	3	120	0.3	12.459841	0.120285	192.080096	187.650738
3	2	3	120	0.4	11.61951	0.249148	216.333686	210.20961
3	2	3	120	0.5	11.095224	0.284248	234.628774	227.382447
3	2	3	120	0.6	16.822688	0.756681	261.387621	266.695551
3	2	3	120	0.7	16.789376	0.766004	262.761264	268.071691
3	2	3	140	0.2	11.755336	1.394253	180.3895	175.97882
3	2	3	140	0.3	12.410259	0.108859	194.164703	188.662631

3	2	3	140	0.4	11.510884	0.178061	217.246595	209.869937
3	2	3	140	0.5	11.085155	0.323894	234.938016	226.383285
3	2	3	140	0.6	16.665625	0.632898	266.802853	271.812179
3	2	3	140	0.7	16.701515	0.50627	264.519267	269.975479
3	3	1	20	0.2	7.011623	0.392325	229.409866	227.348455
3	3	1	20	0.3	7.695312	0.698292	217.887926	216.645414
3	3	1	20	0.4	9.530491	0.728449	192.331677	191.784936
3	3	1	20	0.5	11.205823	1.165511	177.753068	178.100991
3	3	1	20	0.6	16.712991	1.454773	243.177753	246.008683
3	3	1	20	0.7	15.402503	1.143799	249.469714	252.730344
3	3	1	40	0.2	7.039476	0.427986	233.070091	228.562202
3	3	1	40	0.3	7.42655	0.533658	222.818408	220.127386
3	3	1	40	0.4	9.388691	0.670394	192.264263	191.052738
3	3	1	40	0.5	10.499545	0.849249	182.991303	183.902136
3	3	1	40	0.6	16.477605	1.647745	242.69895	248.314977
3	3	1	40	0.7	15.127683	1.52133	251.388812	257.988803
3	3	1	60	0.2	6.84764	0.332642	235.870302	228.978961
3	3	1	60	0.3	7.387855	0.401881	222.125346	217.630264
3	3	1	60	0.4	8.357179	0.799188	205.869851	203.858166
3	3	1	60	0.5	10.060368	1.194249	186.872851	188.217857
3	3	1	60	0.6	16.734371	1.197108	237.319913	245.318012
3	3	1	60	0.7	14.559286	2.012632	253.983014	263.983151
3	3	1	80	0.2	6.765712	0.308689	238.710955	229.31838
3	3	1	80	0.3	7.235209	0.364622	226.387432	219.883354
3	3	1	80	0.4	8.393132	0.865149	205.107838	202.09723
3	3	1	80	0.5	8.248152	1.694033	206.485978	208.839307
3	3	1	80	0.6	15.578566	2.280785	244.853344	255.767102
3	3	1	80	0.7	13.830477	1.510666	258.113149	271.300702
3	3	1	100	0.2	6.730984	0.305418	243.702948	230.843308

3	3	1	100	0.3	7.244308	0.342395	228.945765	220.23484
3	3	1	100	0.4	7.802721	0.630073	215.098996	210.895948
3	3	1	100	0.5	9.329012	1.602192	192.557549	195.633351
3	3	1	100	0.6	16.032532	1.268282	239.298222	252.525922
3	3	1	100	0.7	14.210154	1.393586	251.627635	267.700754
3	3	1	120	0.2	6.677005	0.262005	244.085244	227.970106
3	3	1	120	0.3	7.098341	0.318609	230.988147	220.426993
3	3	1	120	0.4	7.978326	0.777912	214.218376	208.517871
3	3	1	120	0.5	8.689074	1.496775	202.368541	204.989912
3	3	1	120	0.6	15.6652	1.69165	241.601492	257.279604
3	3	1	120	0.7	13.714078	1.38865	254.499287	273.395477
3	3	1	140	0.2	6.493772	0.24356	251.384202	231.08332
3	3	1	140	0.3	7.217228	0.230421	232.504612	218.839237
3	3	1	140	0.4	7.67524	0.364823	222.878686	215.064067
3	3	1	140	0.5	8.311415	1.48965	206.592875	208.553402
3	3	1	140	0.6	15.834062	2.434149	241.218028	258.239519
3	3	1	140	0.7	15.565135	1.325258	239.279979	258.053314
3	3	2	20	0.2	7.105369	0.458979	229.42935	225.289415
3	3	2	20	0.3	7.638804	0.625815	219.146444	216.770623
3	3	2	20	0.4	8.796071	1.0362	199.922268	198.744883
3	3	2	20	0.5	10.276696	1.302809	185.70567	185.8503
3	3	2	20	0.6	16.688522	1.191417	242.836832	246.529204
3	3	2	20	0.7	14.400966	2.007292	260.063684	264.824366
3	3	2	40	0.2	6.818885	0.390394	236.314868	227.788356
3	3	2	40	0.3	7.373821	0.446434	224.199812	219.252082
3	3	2	40	0.4	7.792353	0.665347	214.524513	212.113551
3	3	2	40	0.5	9.162638	1.307602	196.140716	197.595135
3	3	2	40	0.6	14.922787	2.204863	253.802389	262.231191
3	3	2	40	0.7	14.009588	1.715095	259.223039	268.783926

3	3	2	60	0.2	6.744576	0.240854	242.170905	228.780005
3	3	2	60	0.3	7.336913	0.285578	225.916906	217.346329
3	3	2	60	0.4	7.892403	0.697593	215.645992	211.166541
3	3	2	60	0.5	8.386654	1.325279	205.393576	207.402153
3	3	2	60	0.6	15.471979	1.792589	247.026886	258.484331
3	3	2	60	0.7	14.251072	1.663412	252.577751	266.14843
3	3	2	80	0.2	6.631737	0.20282	247.745177	229.446351
3	3	2	80	0.3	7.146261	0.413286	234.272107	221.974997
3	3	2	80	0.4	7.367697	0.394745	224.574457	218.467595
3	3	2	80	0.5	8.054721	1.174298	209.927631	212.183081
3	3	2	80	0.6	15.132913	2.115424	245.505649	260.206786
3	3	2	80	0.7	13.678045	1.504528	254.677982	273.09534
3	3	2	100	0.2	6.565342	0.15179	255.075	230.734475
3	3	2	100	0.3	7.101188	0.269856	236.962258	220.766754
3	3	2	100	0.4	7.074368	0.099363	232.5537	224.663594
3	3	2	100	0.5	7.53871	1.178862	218.052553	221.867599
3	3	2	100	0.6	14.534886	2.118057	248.639746	267.934915
3	3	2	100	0.7	13.603652	1.128148	255.114678	277.715977
3	3	2	120	0.2	6.736655	0.449719	255.554301	224.378378
3	3	2	120	0.3	7.273005	0.1857	241.537556	219.867464
3	3	2	120	0.4	7.112576	0.114138	234.693356	223.732216
3	3	2	120	0.5	7.593712	0.700823	222.425447	224.937523
3	3	2	120	0.6	14.374436	2.279013	247.202343	269.143155
3	3	2	120	0.7	13.755878	1.016505	249.567953	274.594229
3	3	2	140	0.2	7.104782	0.527741	256.698784	220.182332
3	3	2	140	0.3	7.307913	0.169685	245.940148	219.069672
3	3	2	140	0.4	7.176597	0.342407	242.234952	227.589919
3	3	2	140	0.5	14.630209	1.780041	245.029143	267.001278
3	3	2	140	0.6	14.721482	2.091246	244.706844	268.2571

3	3	2	140	0.7	14.138429	1.194629	249.65215	279.156261
3	3	3	20	0.2	7.087945	0.339834	226.729075	226.205192
3	3	3	20	0.3	7.799432	0.561509	214.998768	214.704662
3	3	3	20	0.4	10.014636	0.653987	185.502237	185.391942
3	3	3	20	0.5	11.039997	0.559824	178.710196	179.013952
3	3	3	20	0.6	16.564816	2.574916	245.490952	247.040118
3	3	3	20	0.7	15.585139	1.113059	251.086891	252.778819
3	3	3	40	0.2	7.006851	0.395404	227.70758	226.575882
3	3	3	40	0.3	7.857044	0.267126	212.2027	211.489304
3	3	3	40	0.4	9.281322	0.726121	192.635642	192.380218
3	3	3	40	0.5	11.034728	0.614024	178.341235	179.088581
3	3	3	40	0.6	16.169091	2.407441	248.952164	252.085296
3	3	3	40	0.7	14.973971	2.099463	255.764685	259.189942
3	3	3	60	0.2	6.854535	0.282265	230.480568	228.933747
3	3	3	60	0.3	7.466226	0.266143	218.405866	217.153987
3	3	3	60	0.4	9.279909	0.893893	193.330931	192.940524
3	3	3	60	0.5	9.862411	1.356426	186.316866	187.164776
3	3	3	60	0.6	17.06168	1.547052	239.318484	243.654851
3	3	3	60	0.7	14.937996	2.088114	252.38138	257.432493
3	3	3	80	0.2	6.888301	0.302212	230.323313	228.140884
3	3	3	80	0.3	7.560969	0.353952	217.494254	215.841799
3	3	3	80	0.4	9.310584	0.773593	193.045641	192.353792
3	3	3	80	0.5	9.843494	1.258235	187.378542	188.559928
3	3	3	80	0.6	16.667589	1.276081	241.002876	246.666274
3	3	3	80	0.7	14.495292	1.500081	256.69399	263.486382
3	3	3	100	0.2	6.838971	0.272519	233.654899	230.445539
3	3	3	100	0.3	7.448923	0.336302	217.875989	215.727039
3	3	3	100	0.4	9.167553	1.018629	194.376904	193.313086
3	3	3	100	0.5	8.841248	1.751794	198.986238	200.324796

3	3	3	100	0.6	15.872534	1.850886	246.15269	253.390575
3	3	3	100	0.7	14.364447	1.606355	255.651386	263.907832
3	3	3	120	0.2	6.965307	0.350247	229.522955	225.51093
3	3	3	120	0.3	7.492137	0.367313	217.334758	214.443361
3	3	3	120	0.4	9.115688	0.866003	196.925489	195.570386
3	3	3	120	0.5	8.875458	1.839686	195.750226	197.274796
3	3	3	120	0.6	16.044972	1.893514	243.310509	251.551045
3	3	3	120	0.7	15.423625	2.271086	247.781304	256.7243
3	3	3	140	0.2	7.091312	0.552885	229.248121	224.807141
3	3	3	140	0.3	7.533526	0.284938	217.852959	214.500948
3	3	3	140	0.4	9.337574	0.656405	194.268892	192.54073
3	3	3	140	0.5	17.533018	1.868118	231.695479	240.37703
3	3	3	140	0.6	15.697553	2.743962	247.274748	256.924624
3	3	3	140	0.7	15.820996	1.908801	242.443036	252.3911
3	4	1	20	0	50.0	0.0	166.943292	164.606235
3	4	1	40	0	50.0	0.0	168.056712	162.959959
3	4	1	60	0	50.0	0.0	167.962913	160.036003
3	4	1	80	0	50.0	0.0	169.133199	158.271511
3	4	1	100	0	50.0	0.0	169.728714	155.863603
3	4	1	120	0	50.0	0.0	169.075441	152.181389
3	4	2	20	0	50.0	0.0	167.629872	161.680093
3	4	2	40	0	50.0	0.0	168.736423	156.214166
3	4	2	60	0	50.0	0.0	169.132905	149.946108
3	4	2	80	0	50.0	0.0	170.861681	144.665074

3	4	2	100	0	50.0	0.0	171.316997	138.108032
3	4	2	120	0	50.0	0.0	172.851413	131.945115
3	4	3	20	0	50.0	0.0	167.877139	167.786479
3	4	3	40	0	50.0	0.0	167.887711	167.308166
3	4	3	60	0	50.0	0.0	168.354208	167.275301
3	4	3	80	0	50.0	0.0	168.732594	167.207593
3	4	3	100	0	50.0	0.0	168.054683	166.0492
3	4	3	120	0	50.0	0.0	167.835417	165.224988